Федеральное государственное бюджетное образовательное учреждение высшего образования «Уральский государственный лесотехнический университет»

На правах рукописи

Круглов Артем Васильевич

РАЗРАБОТКА И ИССЛЕДОВАНИЕ МЕТОДИКИ УЧЕТА И АНАЛИЗА ПАРТИИ КРУГЛОГО ЛЕСА С ИСПОЛЬЗОВАНИЕМ ЦИФРОВОЙ ОБРАБОТКИ ИЗОБРАЖЕНИЙ

Специальность 05.21.01 – Технология и машины лесозаготовок и лесного хозяйства

диссертация на соискание ученой степени
кандидата технических наук

Научный руководитель:

кандидат технических наук,
профессор Мехренцев А.В.

Екатеринбург, 2017

Введение 7

Глава 1. Современное состояние рынка решений по автоматизации и информатизации предприятий лесопромышленной отрасли 15

1.1 Анализ использования лесных ресурсов Российской Федерации 15
1.2 Характеристика лесной промышленности Российской Федерации 20
1.3 Особенности учета лесного сырья и готовой продукции 23
1.4 Современное состояние исследований по заявленной тематике 25
1.5 Выводы 27

Глава 2. Методика измерения объема штабеля круглого леса 31

2.1 Анализ существующих методов измерения объемов круглого лесоматериала 31
2.1.1 Поштучное измерение диаметра в верхнем торце 31
2.1.2 Метод концевых сечений 32
2.1.3 Геометрический метод измерения 32
2.1.4 Измерение объема методом гидростатического взвешивания 33
2.1.5 Весовой метод измерения 34
2.1.6 Бесконтактные методы измерения 35
2.1.7 Классификация методов измерения 37
2.2 Описание предлагаемой методики 39
2.3 Выводы 44

Глава 3. Разработка и исследование методов цифровой обработки и анализа изображений для создания алгоритма автоматического выделения торцов бревен на изображении 46

3.1 Обзор научной литературы по методам выделения круглых объектов на изображении 47
3.2 Разработка алгоритма выделения торцов бревен на изображении 51
3.2.1 Детектирование объектов 52
3.2.2 Кластеризация 64

3.2.3 Сегментация 66
3.3 Сравнение результатов с ручным измерением 70
3.4 Выводы 79
Глава 4. Требования к исходным данным 81
4.1 Выбор конфигурации съемки 81
4.2 Калибровка сенсоров 82
4.3 Выводы 84
Глава 5. Измерение объема штабеля круглого леса 85
5.1 Построение модели пакета бревен 85
5.2 Измерение объема штабеля круглого леса 87
5.3 Выводы 91
Глава 6. Программа для мобильного расчета объема штабеля круглого леса 92
6.1 Назначение программы 92
6.2 Описание логической структуры 93
6.2.1 Структура программы 94
6.2.1.1 Модуль основного окна программы 95
6.2.1.2 Модуль настроек программы 95
6.2.1.3 Модуль окна навигатора 95
6.2.1.4 Модуль просмотра и сохранения отчетов 95
6.2.1.5 Модуль алгоритма 95
6.2.1.6 Модуль настроек программы 96
6.2.1.7 Модуль констант 96
6.2.1.8 Модуль методов расчета объема 96
6.2.1.9 Модуль загрузчика изображений 97
6.2.1.10 Модуль математических функций 97
6.2.1.11 Модуль модели 97
6.2.1.12 Модуль настроек модели 97
6.2.1.13 Модуль формирования отчетов 97

6.2.1.14 Модуль настроек и визуализации 98
6.2.1.15 Модуль контроллера 98
6.2.1.16 Модуль графических примитивов 98
6.3 Алгоритм работы программы 98
6.4 Интерфейс программы 100
6.4.1 Главное окно программы 100
6.4.1.1 Главное меню 101
6.4.1.2 Панель инструментов 101
6.4.1.3 Рабочая область 102
6.4.1.4 Строка состояния 102
6.4.2 Окно настроек программы 103
6.4.3 Окно таблиц объемов 104
6.4.4 Окно открытия изображений 105
6.4.5 Окно навигатора 106
6.4.6 Окно справочной информации 106
6.4.7 Окно отчетов 107
6.5 Выполнение программы 109
6.5.1 Установка программы 109
6.5.2 Запуск и выполнение программы 112
6.5.3 Загрузка изображений в программу 112
6.5.4 Калибровка изображений 114
6.5.5 Автоматический поиск бревен 116
6.5.6 Ручное редактирование 119
6.5.7 Получение и анализ результата 121
6.5.8 Завершение работы программы 121
6.6 Сообщения оператору 121
6.7 Выводы 124

Глава 7. Результаты тестирования методики мобильного расчета объема штабеля круглого леса 125

7.1 Программа и методика исследовательских испытаний программы для мобильного расчета объема штабеля круглого леса 125
7.1.1 Требования к средствам проведения испытаний 125
7.1.2 Требования к условиям проведения испытаний..... 126
7.1.3 Требования к подготовке изделия к испытаниям..... 126
7.1.4 Требования к персоналу, осуществляющему подготовку к испытаниям и испытания 127
7.1.5 Требования безопасности 127
7.1.6 Программа испытаний..... 127
7.1.7 Режимы испытаний..... 128
7.1.7.1 Порядок испытаний..... 128
7.1.7.2 Ограничения и другие указания, которые необходимо выполнять на всех или на отдельных режимах испытаний..... 128
7.1.7.3 Условия перерыва, аннулирования и возобновления испытаний на всех или на отдельных режимах. 129
7.1.8 Методы испытаний 129
7.1.8.1 Проверка по п. 1. Программы..... 129
7.1.8.2 Проверка по п. 2 Программы..... 130
7.1.8.3 Проверка по п. 3 Программы..... 130
7.1.8.4 Проверка по п. 4 Программы..... 131
7.2 Результаты испытания программы по пункту 1 133
7.2.1 Замечания и рекомендации..... 134
7.3 Результаты испытания программы по пункту 2 135
7.4 Результаты испытания программы по пункту 3 137
7.5 Результаты испытания программы по пункту 4 138
7.6 Выводы..... 141
Заключение 145
Список литературы 149
Приложение А 159

Приложение Б 170

Приложение В 189

ВВЕДЕНИЕ

Особенностью лесной отрасли является то, что большинство технологических процессов плохо поддаются автоматическому контролю из-за удаленности производства от необходимой информационной инфраструктуры, и в настоящее время контроль таких процессов осуществляется вручную, без применения информационных систем. Эту проблему можно решить путем внедрения специализированных инструментов контроля на основе машинного зрения и обработки изображений.

Основной целью работы является создание методики и инструментария для оперативного мобильного учета перемещения лесозаготовок в рамках бизнес-логики лесопромышленного предприятия. При этом одним из основных научных направлений является разработка и модификация существующих алгоритмов автоматического выделения и анализа объектов на их изображениях применительно к задаче анализа лесоматериалов.

Актуальность темы исследования. Рациональная эксплуатация лесных ресурсов является одним из средств социально-экономического развития субъектов Российской Федерации, показателем качественного управления лесными ресурсами. Лесной кодекс РФ, принятый в январе 1997 г, предоставил юридическим и физическим лицам (в том числе иностранным) возможность пользования лесным фондом на срок до 49 лет. Этот законодательный акт резко повысил требования к оценке лесного фонда, порядку ведения лесного хозяйства, квалификации лесопользователей. Разнообразие применяемых технологий и систем лесозаготовительной техники, большой объем продажи лесоматериалов за рубеж, а также внедрение сертификации лесов и лесопродукции повышает требования к вопросам точности учета леса на корню и готовой продукции. Сегодня лесной сектор экономики переживает серьезные структурные изменения, сопровождающиеся введением инноваций на управленческом и технологическом уровнях. Это, по мнению экспертов, позволит создать условия для преодоления экономического кризиса и устойчивого развития.

Основным тезисом государственной программы развития лесного хозяйства до 2020 г является необходимость модернизации отрасли в целом. На настоящее время обозначен ряд проблем, которые мешают эффективному лесоуправлению: неточности при оценке объемов доступного сырья, недостаточная доля применения в лесхозе новых информационных технологий, нехватка квалифицированных кадров и производственных мощностей. Также важным фактором, определяющим направление развития отрасли, стал принятый 28 декабря 2013 года Федеральный закон № 415-ФЗ «О внесении изменений в Лесной кодекс Российской Федерации и Кодекс Российской Федерации об административных правонарушениях», который обязует юридических лиц декларацию в электронном виде о совершенных ими сделках с древесиной в ЕГАИС «Учет древесины и сделок с ней» не позднее одного дня после заключения соответствующего договора. Таким образом, заготовители, продавцы и перевозчики круглых лесоматериалов приходят к необходимости внедрения на производстве современных методов учета лесоматериалов, обеспечивающих оперативную, достоверную информацию о технологическом процессе, составляющую основу документооборота. Без достоверных данных о количестве и качестве круглых лесоматериалов невозможны как эффективное управление лесным комплексом в целом, так и государственное регулирование их оборота

Разработка методики автоматического расчета объема партий круглого леса с использованием специализированного программного продукта на основе фотограмметрии позволит:

– осуществлять оперативный сквозной контроль производственных процессов, связанных с перемещением партий круглого леса (приемка/отгрузка, транспортировка на сортиментовозе, отправка на переработку),

– автоматизировать документооборот и упростить процедуры отчетности для предприятия.

Мобильность и скорость работы данного решения позволит проводить контрольные замеры в местах заготовки леса, при погрузке на сортиментовоз, в момент отгрузки-приемки и при отправке на переработку, т.е. осуществлять учет

сортимента на каждом этапе его жизненного цикла. Преимущества предлагаемого решения по сравнению с существующими подходами, такими как ручной поштучный или штабельный учет, взвешивание на автомобильных весах, гидростатический метод или применение лазерных сканеров для группового учета, заключаются в следующем:

- скорость получения данных,
- точность и верифицируемость результатов,
- отсутствие привязки к месту измерения и зависимости от внешних факторов (наличие связи и т.п.),
- исключение человеческого фактора из процесса измерения.

В результате достигается высокая эффективность технологических процессов, связанных с учетом круглого леса, что дает возможность оптимизации и информатизации документооборота в целом, повышения оперативности и достоверности анализа и планирования деятельности лесопромышленного предприятия в целом.

Заинтересованность предприятий лесопромышленного комплекса (ЛПК), а также контролирующих структур (ГИБДД, таможенные органы) в разработках такого рода подтверждается, в частности, письмами с согласием о сотрудничестве в проведении испытаний и готовностью приобрести разрабатываемые продукты в случае соответствия их заявленным характеристикам. Содержание и результаты диссертационной работы соответствуют задаче совершенствования информационного обеспечения планирования и управления лесами, методов инвентаризации и мониторинга лесов, заявленной в Стратегии развития лесного комплекса Российской Федерации на период до 2020 года.

Степень разработанности темы. Вопросам разработки и анализа методик измерения круглых лесоматериалов посвящены работы Батурина К.В., Воскобойникова И.В., Гончарова М.Г., Мильцина А.Н., Самойлова А.Н., Солдатова А.В., Старикова А.В.,, Суровцевой Л.В., Шиловой Е.Г. и др. В работе Бита Ю.А. рассмотрен фотометрический метод измерения, который отличают высокие требования к количеству и сложности оборудования. Автоматические

измерительные системы на основе автокубатурников представлены в работе Петровского В.С. В работах Санникова С.П. представлен метод мониторинга транспортных потоков древесины на основе RFID меток. Распознаванию торцов бревен на изображениях посвящены работы Князя В.А., а также зарубежных авторов Хербона С., Гутзейта Е., Галсгаарда Б.

В рассмотренных работах предложенные подходы не отвечали задаче эффективного измерения партии леса в условиях ограниченного времени, соответствующего методам поштучного измерения круглых лесоматериалов ГОСТ 32594-2013, либо же предложенные методика и ее реализация не обеспечивали необходимый уровень точности и достоверности.

Диссертация представляет собой законченное научное исследование, включающее в себя изучение состояния проблемы и постановку цели и задач, теоретический анализ возможности автоматизации процессов измерения и учета партий круглых лесоматериалов с использованием мобильных средств измерений на основе фотограмметрического анализа изображений торцов штабеля и практическую реализацию методики на основе алгоритма автоматического распознавания торцов бревен в партии.

Цель научного исследования. Совершенствование работы лесопромышленного предприятия за счет повышения эффективности процесса сбора и анализа данных о движении сырья (сортимента) в пунктах опорного учета – в местах заготовки, приемки и отгрузки круглого леса.

Задачи научного исследования:

1. Провести анализ существующих методов и методик измерения круглого леса, выявить их достоинства и недостатки, допустимые погрешности измерений.
2. Провести анализ и выбрать методы и алгоритмы обработки и анализа данных, оптимальные для задачи автоматической обработки и анализа изображений.
3. Разработать алгоритм автоматического измерения партии круглого леса на основе фотограмметрии и интеллектуальной обработки изображений.

4. Разработать методологию учета сортимента на основе разработанного алгоритма.

5. Провести экспериментальную оценку разработанного решения в условиях промышленного производства и установить погрешности измерений по предложенной методике.

Объект исследования – процесс сквозного учета круглого леса на лесопромышленных предприятиях как объект автоматизации.

Предмет исследования – методика и алгоритм оперативного автоматизированного измерения объема партии круглого леса на основе фотограмметрического анализа изображений.

Научная новизна состоит в разработке алгоритма обработки изображений, осуществляющего автоматическое распознавание и выделение торцов бревен на изображении на основе алгоритма поиска объектов с радиальной симметрией с применением методов фильтрации, кластеризации и сегментации объектов интереса. Применение данного алгоритма лежит в основе предложенной методики оперативного измерения объема партии круглого леса на основе фотограмметрического анализа изображений путем построения 3D-модели штабеля бревен по одной или двум проекциям его торцов.

В результате проведения исследования разработаны методика и алгоритм автоматического расчета объема партии круглого леса на основе фотограмметрии, получены патент на изобретение методики измерения кубатуры круглого леса (Патент № 2553714 от 22 мая 2015 г), свидетельство о регистрации программы «FoRest» – авторского программного продукта для мобильного учета партии круглого леса (Свидетельство о регистрации программы для ЭВМ № 2014611152).

Включенный в диссертацию материал отражает личный вклад автора в создание методики и алгоритма измерения объема партии круглого леса, а также в разработку программы для мобильного учета партии круглого леса. Кроме того, автор диссертации разработал методическое и организационное обеспечение, принимал непосредственное участие в процессах проектирования, разработки, тестирования и внедрения практических результатов работы.

Теоретическая и практическая значимость исследования. Теоретическая значимость исследования состоит в разработке эффективной методики расчета объема партии круглого леса фотограмметрическим способом на основе поштучного метода измерения, отличающейся от существующих методик тем, что расчет выполняется автоматически, без участия оператора, при этом строится 3D-модель штабеля бревен по изображению его торцов, и по данной модели определяется ориентация бревен в штабеле, что повышает точность расчетов партии круглого леса. Практическая значимость исследования заключается в разработке и внедрении на ряде предприятий программы для мобильного расчета объема партии круглого леса, в которую входят авторский алгоритм автоматического выделения торцов бревен и пользовательский интерфейс, включающий инструменты для редактирования результатов работы алгоритма оператором.

Методы исследований. При разработке алгоритма и программы для расчета кубатуры круглого леса использовались методы цифровой обработки сигналов, методы обнаружения и распознавания объектов на изображении, методы теории графов, методы объектно-ориентированного анализа, системный анализ, математическое моделирование, математическая статистика, методы разработки программных средств. Проверка работоспособности предложенного алгоритма интеллектуальной обработки изображений выполнялась с помощью экспериментальных исследовании в условиях промышленного производства.

Научные положения, выносимые на защиту:

1. Алгоритм автоматического выделения торцов бревен на изображении на основе метода поиска объектов с радиальной симметрией, дополненного функцией фильтрации выделенных объектов, их кластеризацией и сегментацией с целью уточнения контуров выделенных торцов.
2. Методика и алгоритм расчета объема партии круглого леса на основе фотограмметрии с учетом ориентации бревен в штабеле, позволяющие проводить количественный анализ партии леса с помощью мобильных устройств

посредством интеллектуальной обработки изображений одного или двух торцов штабеля.

3. Разработанное программное обеспечение средства измерения для мобильного расчета объема партии круглого леса, в которое входят авторский алгоритм автоматического детектирования торцов бревен и пользовательский интерфейс, включающий инструменты для редактирования результатов работы алгоритма оператором.

Соответствие содержания диссертации паспорту специальности. Основные результаты диссертационной работы соответствуют пункту 3 «Разработка операционных технологий и процессов в лесопромышленном и лесохозяйственном производствах: заготовительном, транспортном, складском, обрабатывающем, лесовосстановительном и др.» и пункту 9 «Автоматизация управления машинами, выбор систем учета лесопродукции, эргономика и безопасность условий труда» паспорта специальности 05.21.01 «Технология и машины лесозаготовок и лесного хозяйства» (технические науки).

Реализация результатов работы. Основная часть исследований и разработок по тематике диссертации выполнялась с целевой направленностью в рамках госбюджетных работ и грантов (контракты №8888р/14418 от 11.04.2011 г, №11840р/14418 от 05.04.2013 г, №816ГС3/14418 от 30.11.2015 г). Создан программный продукт для мобильной оценки объема партии круглого леса. Результаты исследований внедрены на ряде предприятий Свердловской области: ООО «АРМ-Рус», ООО «ИТК-Дизель», ООО «Энкон-сервис» (акт внедрения, лицензионный договор), ИП Козьменко, ИП Федореев (лицензионный договор).

Апробация работы. Основные положения и результаты диссертационной работы докладывались на 10 научных конференциях, в том числе: 10th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - VISIGRAPP -2015 (Берлин, Германия, 2015); 12th International Conference on Informatics in Control, Automation and Robotics ICINCO 2015 (Кольмар, Франция, 2015); IV-й Всероссийской отраслевой научно-практической конференции студентов, молодых ученых и специалистов

«Перспективы развития техники и технологий в целлюлозно-бумажной промышленности» (Пермь, 2016); First International Workshop on Pattern Recognition IWPR 2016 (Токио, Япония, 2016); International Conference on Digital Image Computing: Techniques and Applications DICTA 2016 (Голд Коаст, Австралия, 2016); International Conference on Applied Mathematics and Computer Science ICAMCS 2017 (Рим, Италия, 2017). Кроме того, результаты работы были представлены на ряде отраслевых промышленных выставок: INNOPROM 2012-13 (Екатеринбург), INTERFORST-2014 (Мюнхен, Германия), LESPROM-URAL Professional 2014-15 (Екатеринбург), ELMIA Wood 2017 (Йенчепинг, Швеция).

Разработанная методика учета и анализа партий круглого леса была представлена и одобрена на выездном заседании «Уральского союза лесопромышленников» в п. Коуровка (Свердловская область) в 2015 г.

Публикации. По результатам исследований опубликовано 18 работ, из них 3 в рецензируемых изданиях, рекомендуемых ВАК РФ, 7 – в изданиях, индексируемых в Scopus/WoS.

Структура и объём работы. Диссертация состоит из введения, семи глав, заключения, трех приложений и списка литературы из 96 наименований. Общий объем работы составляет 261 страницу машинописного текста (из них 103 страницы приложения), содержит 69 рисунков и 16 таблиц.

ГЛАВА 1. СОВРЕМЕННОЕ СОСТОЯНИЕ РЫНКА РЕШЕНИЙ ПО АВТОМАТИЗАЦИИ И ИНФОРМАТИЗАЦИИ ПРЕДПРИЯТИЙ ЛЕСОПРОМЫШЛЕННОЙ ОТРАСЛИ

1.1 Анализ использования лесных ресурсов Российской Федерации

Леса занимают около 69% территории суши Российской Федерации с внутренними водами и являются важнейшим природным комплексом. Лес обладает двойственной природой возобновляемого природного ресурса, который не только удовлетворяет множественные экономические потребности промышленности в сырье и общества в продовольствии, энергии и жилье, но и является важнейшим средообразующим и средозащитным фактором.

По данным Государственного лесного реестра (ГЛР) [1] на 01.01.2016 г., общая площадь земель Российской Федерации, на которых расположены леса, составила 1184,1 млн га, в том числе площадь земель лесного фонда – 1146,3 млн га, из них площадь резервных лесов (расположенных только в Дальневосточном и Сибирском федеральных округах) составляет 268,5 млн га, а площадь защитных лесов – 279,1 млн га (Таблица 1.1).

Лесные ресурсы – стратегическое конкурентное преимущество лесного комплекса России в мировой экономической системе, но при объеме запасов древесины в 84 млрд м3 – на Россию приходится лишь 5,5% мирового объема лесозаготовки, 3% мировой торговли лесоматериалами. Обладая пятой частью мировых запасов лесов и имея огромный потенциал для освоения лесных ресурсов, Российская Федерация существенно уступает развитым зарубежным странам по уровню заготовки древесины (5-е место после США, Индии, Китая, Бразилии) [2].

Таблица 1.1 – Площадь земель фонда лесного фонда, на которых расположены леса, тыс. га

Федеральный округ	Всего	Площадь резервных лесов	Площадь защитных лесов, в том числе				
			всего	расположенные на ООПТ	расположенные в водоохранных зонах	выполняющие функции защиты – всего	ценные леса – всего
Российская Федерация	1146297,40	1146297,40	1146297,40	1146297,40	1146297,40	1146297,40	1146297,40
Дальневосточный	496591,80	496591,80	496591,80	496591,80	496591,80	496591,80	496591,80
Крымский	0,00	0,00	0,00	0,00	0,00	0,00	0,00
Приволжский	39049,00	39049,00	39049,00	39049,00	39049,00	39049,00	39049,00
Северо-Западный	112879,80	112879,80	112879,80	112879,80	112879,80	112879,80	112879,80
Северо-Кавказский	1723,90	1723,90	1723,90	1723,90	1723,90	1723,90	1723,90
Сибирский	358475,20	358475,20	358475,20	358475,20	358475,20	358475,20	358475,20
Уральский	112100,40	112100,40	112100,40	112100,40	112100,40	112100,40	112100,40
Центральный	22724,00	22724,00	22724,00	22724,00	22724,00	22724,00	22724,00
Южный	2753,30	2753,30	2753,30	2753,30	2753,30	2753,30	2753,30

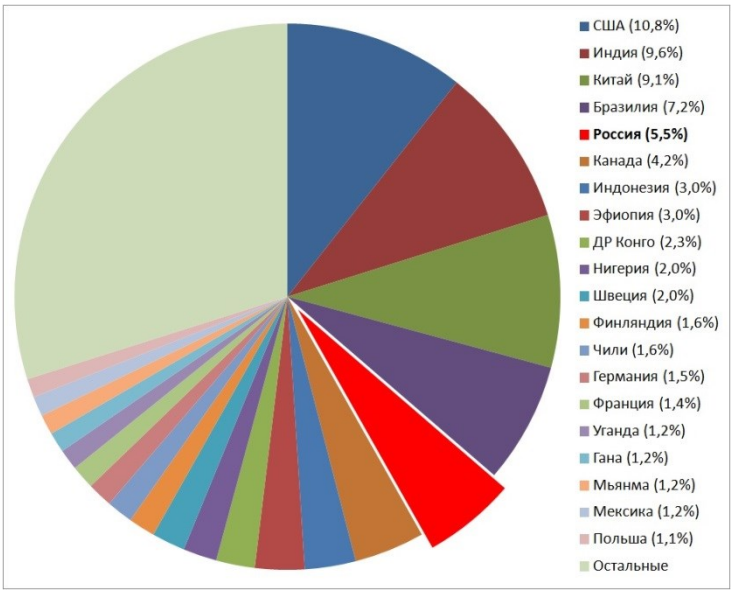


Рисунок 1.1 – Распределение мировых объемов учтенной заготовки древесины по крупнейшим странам-лесозаготовителям в 2015 году по данным Продовольственной и сельскохозяйственной организации ООН (FAO)

По информации Федеральной службы государственной статистики индекс целлюлозно-бумажного производства и древесины в России в январе-ноябре 2016г. по сравнению с январем-ноябрем 2015г. составил 101,8%, при этом обработка древесины и производство изделий из дерева показала рост на 2,4% за 11 месяцев и 2,5% по оценке за 12 месяцев 2016 года [3].

График на Рисунке 1.2 демонстрирует тенденцию к росту производства после значительного спада в 2015 г.

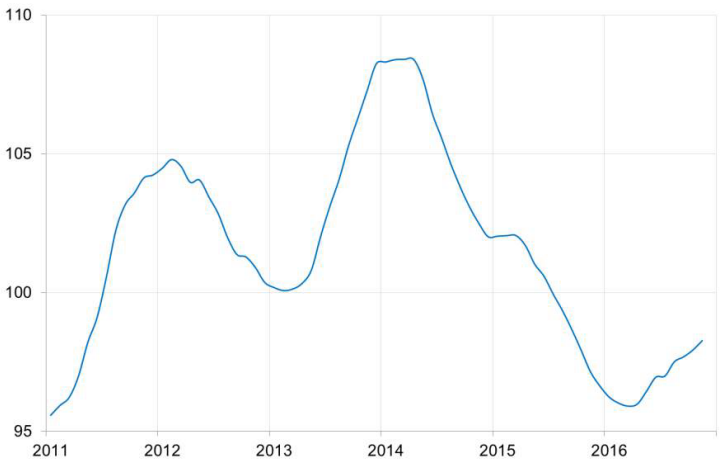


Рисунок 1.2 – Обработка древесины и производство изделий из дерева, очищ/сезон., 2011 г. =100

Основные тренды производства целлюлозно бумажной промышленности и древесины могут быть сформулированы следующим образом.

– Производство основных видов продукции в ноябре 2016 года показало положительную динамику: в производстве клееной фанеры, прирост составил 8,8% г/г, изготовление плит древесноволокнистых из древесины или других одревесневших материалов в ноябре показывает хорошую динамику прироста 6,3% г/г. Данный прирост можно объяснить продолжением обработки большого количества заготовленной ранее древесины.

– Несмотря на номинальное сокращение индекса целлюлозно-бумажного производства, издательской и полиграфической деятельности в ноябре (-0.2% г/г), за период с начала года сохраняется положительная динамика (2.0 % г/г).

– Согласно данным Росстата, производство целлюлозы, древесной массы, бумаги, картона и изделий из них в ноябре увеличилось на 4.0 процента. Стабильный рост целлюлозно-бумажной промышленности можно объяснить общим ростом производства в лесной промышленности и увеличением притока сырья для создания целлюлозно-бумажных продуктов.

Графики на Рисунках 1.3-1.4 показывают значительное увеличение объемов заготовок леса за 2016г.

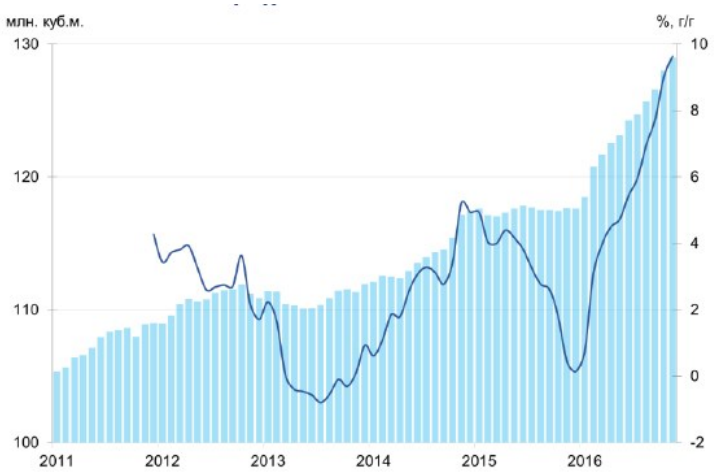


Рисунок 1.3 – Объемы заготовки древесины

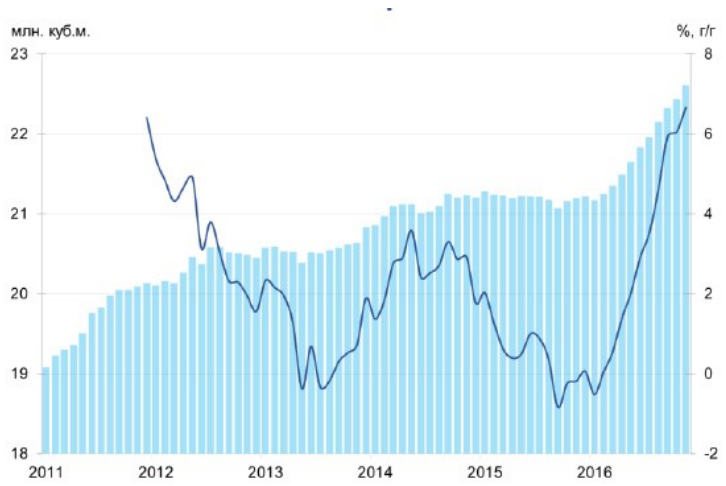


Рисунок 1.4 – Объемы заготовки пиломатериалов

В январе-сентябре 2016г. было выпущено:

– Лесоматериалов, продольно распиленных или расколотых, разделенных на слои или лущеные, толщиной более 6 мм; шпал железнодорожных или трамвайных деревянных, непропитанных – 17,1 млн.м3, что составило 104,5% к январю-сентябрю 2015г.

– Фанеры клееной, состоящей только из листов древесины – 2,8 млн.м3 (102,3% к январю-сентябрю 2015г).

– Плит древесностружечных и аналогичных плит из древесины и других одревесневших материалов – 396 млн.усл.м2 (104,4% к январю-сентябрю 2015г).

– Плит древесноволокнистых из древесины или других одревесневших материалов – 128 млн.штук (106,1% к январю-сентябрю 2015г).

– Целлюлозы древесной и целлюлозы из прочих волокнистых материалов – 6,1 млн.тонн (105,6% к январю-сентябрю 2015г).

– Бумаги – 3,9 млн.тонн (103,4% к январю-сентябрю 2015г).

– Картона – 2,5 млн.тонн (107,3% к январю-сентябрю 2015г).

– Бумаги или картона двухслойных гофрированных – 23,5 млн.м2 (103,5% к январю-сентябрю 2015г).

– Блоков дверных в сборе (комплектно) – 8,1 млн.м2 (86,9% к январю-сентябрю 2015г).

1.2 Характеристика лесной промышленности Российской Федерации

Промышленная отрасль, предприятия которой занимаются заготовкой и обработкой дерева, называется лесной промышленностью либо лесным комплексом. Она является одной из самых старых промышленных отраслей и имеет сложную структуру. Продукция лесного комплекса широко используется во многих отраслях промышленности, строительстве, сельском хозяйстве, полиграфии. Запасы дерева в нашей стране составляют более восьмидесяти миллиардов кубометров, для использования подходят более сорока миллиардов кубометров [4].

Каждая часть структуры лесной отрасли отвечает за одну из стадий обработки сырья из древесины. Структура лесной промышленности такова:

– Лесозаготовительная отрасль, в которую входят заготовка древесины, подсечка леса (добывание живицы и заготовка пневого осмола), сплав бревен, деятельность по передаче древесины с одного типа транспорта на другой, применение неценных древесных пород и отходов (лесопилка, выпиливание шпал изготовление щепы, досок для тары). Она является самой крупной отраслью промышленности по обработке леса.

– Деревообрабатывающая отрасль.

– Целлюлозно-бумажная отрасль, которая механически и химически обрабатывает древесное сырье.

– Лесохимическая отрасль, которая перерабатывает сырье из древесины сухим способом, занимается углежжением, созданием канифоли и скипидара. К данной отрасли относится изготовление лака, эфира, пластмассы, ненатуральных волокон, гидролиз (создание этила, дегтя, скипидара из отходов при изготовлении целлюлозно-бумажных изделий).

Лесная и деревообрабатывающая промышленность России условно делятся на следующие группы:

– создание пиломатериалов и предметов мебели (механическая обработка);

– лесохимическая промышленность и создание целлюлозно-бумажных продуктов (химический способ обработки).

Промышленные предприятия, относящиеся к лесной и деревообрабатывающей промышленной отрасли, занимаются:

– заготовкой древесного материала;
– обработкой древесного материала;
– лесохимической промышленной обработкой лесного сырья;
– выпуском целлюлозно-бумажных изделий.

Названные фабрики и заводы производят круглый лес, доски, разные деревянные предметы, лесохимическую продукцию и бумагу.

На территории нашего государства преобладают хвойные деревья, они более ценны для промышленности, чем деревья с листьями. В РФ леса территориально произрастают неравномерно. Больше всего лесов в нескольких областях: в Северной, Уральской, Волго-Вятской, Дальневосточной и Сибирской.

Сегодня лесопромышленный сектор переживает серьезные структурные изменения, сопровождающиеся введением инноваций на управленческом и технологическом уровнях. Это, по мнению экспертов, позволит создать условия для преодоления экономического кризиса и долгосрочного развития. Структурные изменения обусловлены 4 основными причинами:

1. Уменьшение спроса во время кризиса, которое способствовало рационализации производственных мощностей. Сегодня руководители лесоперерабатывающих предприятий стремятся к сокращению издержек, включая затраты на персонал и оплату труда. Автоматизированные системы управления производственными процессами находят все более широкое применение.
2. Доминирующее положение политик, связанных с изменением климата и защитой окружающей среды. Они способствуют быстрому увеличению производства и потребления деревянного топлива как источника энергии.
3. Глобализация рынка леса и продуктов лесной промышленности, в том числе появление такого важного игрока на рынке, как Китай, который сегодня является

как одним из крупнейших экспортеров леса, так и крупнейшим импортером бумаги и других продуктов лесопромышленности.

4. Международный контроль происхождения леса с целью обеспечения производства, отвечающего требованиям законодательства и устойчивого развития.

Эксперты рынка сходятся во мнении, что единственный способ преодоления стагнации для лесопромышленного сектора – внедрение в производство современных информационных технологий. «Оборудование и техника лесной промышленности устарели, а износ основных фондов, эксплуатирующихся еще со времен СССР, превысил 60%»,— констатирует Лариса Иршинская, руководитель направления «Специальные проекты» Cornerstone. По ее словам, российские технологии, внедренные более 40 лет назад, уступают западным на мировых рынках из-за высоких издержек при производстве». Усовершенствование производства в условиях конкуренции необходимо. Устаревшие предприятия, использующие неэффективные технологии, переживают трудности. Из-за слишком больших издержек и перегрузок производства в 2013 году с кризисом столкнулись ОАО «Кондровская бумажная компания», ОАО «Селенгинский ЦКК», ООО «Енисейский ЦБК», отмечает президент Лиги переработчиков макулатуры РАО Бумпром Андрей Гурьянов [5].

Лесозаготовительная и лесоперерабатывающая отрасли РФ работают крайне неэффективно. По данным Министерства природных ресурсов, использование расчетной лесосеки составило в 2015 г. лишь 29,4% [6]. Основным тезисом государственной программы развития лесного хозяйства до 2020 года является необходимость модернизации отрасли в целом. Также, обозначен ряд проблем, которые мешают эффективному лесоуправлению. Среди них неточности при оценке объемов доступного сырья, недостаточная доля применения в лесхозе новых информационных технологий, недостаток квалифицированных кадров и производственных мощностей [7].

1.3 Особенности учета лесного сырья и готовой продукции

По поручению Правительства РФ Рослесхоз с участием других ведомств разработал законопроект «О государственном регулировании оборота круглых лесоматериалов» [8]. Основными задачами этого законопроекта являются:

– восстановление в России обязательного учета круглых лесоматериалов;

– введение государственного регулирования оборота круглых лесоматериалов путем декларирования экономическими субъектами (предприятиями, ИП) проводимых с лесоматериалами операций через Информационную систему.

Согласно законопроекту, заготовители, продавцы и перевозчики круглых лесоматериалов должны будут подавать декларацию в электронном виде о совершенных ими сделках с этими материалами не позднее одного дня после заключения соответствующего договора (заготовители – также и за день до перевозки очередной партии). В декларации указываются, помимо прочего, сведения об объеме передаваемых круглых лесоматериалов, их породном и сортиментном составе.

Также, согласно ст. 11 п. 1 законопроекта, «Лица, осуществляющие в соответствии с лесным законодательством заготовку древесины, за исключением граждан, осуществляющих заготовку древесины для собственных нужд, и лица, осуществляющие оборот круглых лесоматериалов, обязаны осуществлять учёт круглых лесоматериалов при обороте круглых лесоматериалов до их обработки, переработки, изготовления из них продукции или вывоза из Российской Федерации» и ст. 11 п. 3 «Круглые лесоматериалы учитываются по объему, весу, числу. Объем лесоматериалов определяется в плотной или складочной мере».

Принятие этого законопроекта приводит лесопромышленные предприятия к необходимости внедрения на производстве современных методов учета лесоматериалов, обеспечивающих оперативную, достоверную информацию о технологическом процессе, составляющую основу документооборота. Без достоверных данных о количестве и качестве круглых лесоматериалов

невозможны как эффективное управление лесным комплексом в целом, так и государственное регулирование их оборота [9].

Другим фактором является сезонность лесозаготовительного производства, что вынуждает предприятия создавать и хранить значительные запасы круглых лесоматериалов. На крупных целлюлозно-бумажных и деревообрабатывающих комбинатах, а также лесных терминалах запасы складируемой древесины достигают сотен тысяч кубометров. Без достоверного и непрерывного учета лесоматериалов невозможно осуществлять эффективное управление технологическим процессом [10]. На Рисунке 1.5 представлено количество и расположение пунктов учета лесоматериалов на среднестатистическом лесозаготовительном предприятии.

Операции сортировки и учета объемов лесоматериалов и в настоящее время, несмотря на возрастающую инновационную активность лесопромышленных предприятий, зачастую выполняются с применением ручного труда, что обуславливает:

– увеличенный травматизм;
– большие трудозатраты;
– высокая себестоимость;
– низкая производительность труда.

Особенно характерна данная ситуация для малых и средних предприятий, где внедрение автоматических программно-аппаратных комплексов для учета круглого леса (например Система измерения объема круглых лесоматериалов на автотранспорте «СканТрек», ООО «Интерфейс») является экономически необоснованным из-за их высокой стоимости. В свою очередь, доступные решения для малых предприятий, позволяющие проводить быструю, качественную и достоверную оценку объемов лесоматериалов, на рынке до сих пор не представлены, что приводит к необходимости ручных методов измерения.

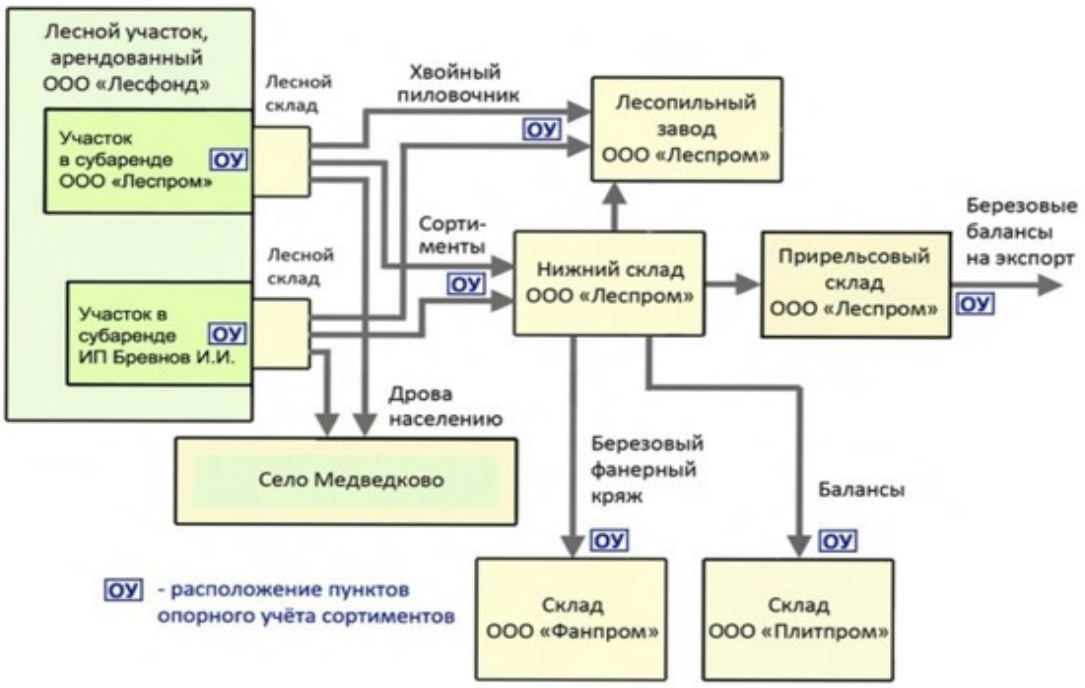


Рис 1.5 – Пример расположения пунктов опорного учета сортиментов

В 2008 году на своей осенней сессии Общества по измерениям лесоматериалов сотрудником компании Woodtech Христианом Пассотом был представлен доклад о различных методиках измерения балансов по всему миру. Несмотря на определенную субъективность автора, материалы презентации дают общее представление о состоянии дел в лесопромышленной отрасли: отсутствие достоверного, исключающего возможность обмана и обеспечивающего высокую точность способа контроля и учета круглого леса в местах его заготовки, хранения, переработки и при транспортировке. Подробнее с материалами презентации можно ознакомиться в Приложении А.

1.4 Современное состояние исследований по заявленной тематике

Разработка систем автоматизированного контроля и учета является одним из наиболее востребованных направлений научных исследований и разработок. При этом с развитием технологий происходит постоянная смена трендов, требований к функциональным особенностям данных систем. Так, в настоящее время наиболее перспективные направления разработки систем контроля заключаются в следующем.

– Self- service аналитика. Идея Self-service состоит в том, что любой бизнес-пользователь может анализировать необходимые ему данные для принятия рутинных решений. Правда, это и подразумевает, что такой пользователь имеет возможность опираться на корректные данные, причем эти данные ему доступны, а формат их использования достаточно прост для каждого. Кроме того, у пользователя должно быть решение, позволяющее создавать аналитические запросы к указанному массиву данных. Когда все эти условия соблюдены, компании получают в действительности рабочий инструмент анализа и принятия решений.

– Неструктурированные данные. К данной категории относятся визуальные данные. В 2011 и 2012 годах появилось множество инструментов для работы с неструктурированным данными, так что анализ такой информации становится одним из главных приоритетов. Причем обработка все чаще происходит в реальном времени.

– Развитие визуальной аналитики. Визуальная аналитика далеко не новость на мировом рынке BI, но на протяжении долгих лет она была компонентом сложных производительных аналитических платформ, а не инструментом массового анализа. Постепенно как вендоры, так и сами пользователи осознали, что визуализация данных может помочь любому из них наилучшим образом представлять и понимать данные. Так что теперь визуальные инструменты доступны массовым пользователям, а не только узкому кругу бизнес-менеджеров и экспертов.

– Мобильная аналитика. Сегмент мобильной аналитики переживает бурное развитие. Уже сегодня значительное число сотрудников используют планшеты для быстрого доступа к данным в своей непосредственной работе. Взаимодействие пользователей с мобильными данными все более тяготеет к аналитике и становится все более интерактивным.

Задача автоматического выделения объектов – крайне популярное направление в анализе и обработке изображений. Ведущие базы научных статей – Web of Science

и Scopus – приводят следующие статистику по количеству публикаций по данной тематике за последние 10 лет:

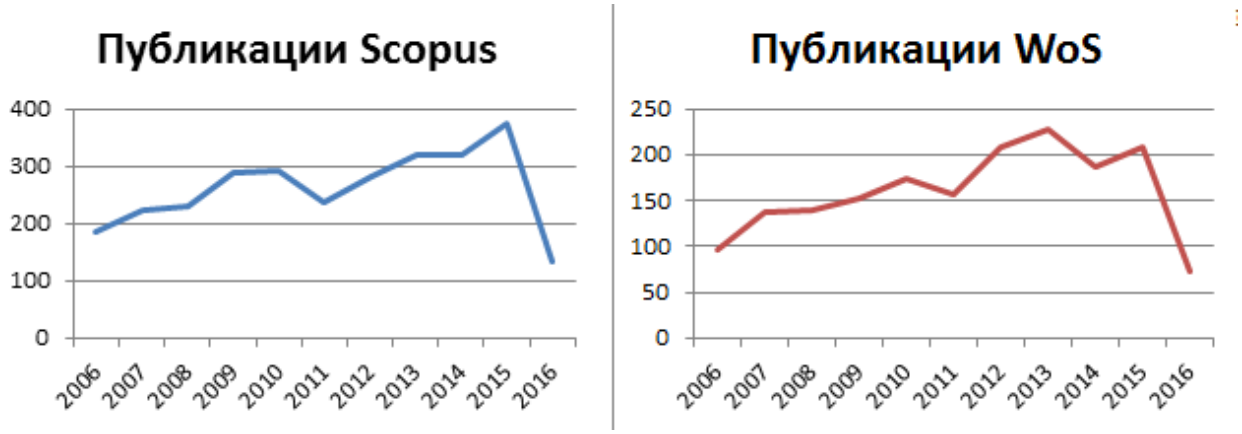


Рисунок 1.6. – Количество публикаций по тематике проекта в Web of Science и Scopus

При этом количество статей, посвященных задаче автоматического выделения бревен, невелико – 16 статей в базе WoS и 8 в Scopus по запросу «log cut detection».

Материалы современных исследований и технологических новинок в области измерения древесины представляются на ежегодной встрече Общества по измерениям древесины (Timber Measurement Society). На последней конференции, прошедшей 6-8 апреля 2016, представлены презентации аналогичных проектов, запущенных в 2015-2016 г.г. в ряде стран – Эстонии, Германии, Бельгии. Это говорит о крайне высокой актуальности данной идеи на общемировом уровне.

1.5 Выводы

Типовые процессы приема, передвижения и производства лесоматериала на нижнем складе связаны с операциями погрузки, разгрузки, измерения и производства. В данных операциях принимают участия следующие специалисты:

– мастера участков;
– водители;
– операторы погрузки;
– операторы разгрузки;

Прием лесоматериала осуществляется после операций вывозки – перемещение сортиментов или хлыстов от погрузочного пункта или лесовозной дороги до склада хранения или переработки (Рисунок 1.7).

Автоматизированный учета и анализа партий круглого леса позволяет аккумулировать данные о производственных процессах на ключевых производственных участках предприятия:

а) Учет поступления лесоматериалов на нижний склад в результате деятельности производственных бригад вырубки леса на декларируемых лесосеках и операций транспортировки леса;

б) Учет операций переработки лесоматериала на НС связанных с передвижением и разделкой лесоматериала;

в) Выбытие лесоматериала с нижнего склада в разрезе основания и направления выбытия с учетом деятельности бригад погрузки.

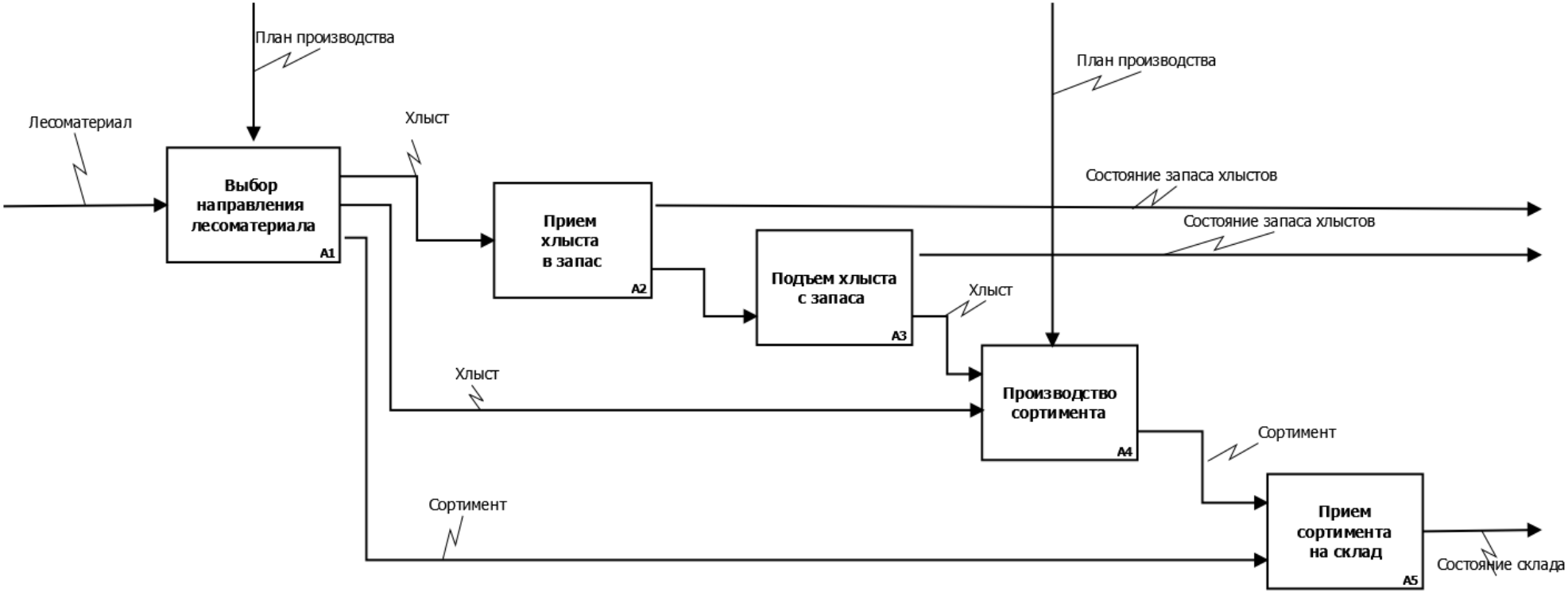


Рисунок 1.7 – Типовой процесс приема и производства лесоматериала

Согласно статистическим данным, по другим отраслям промышленности (металлургическая, горная, нефтегазодобывающая) внедрение систем автоматизации систем контроля и учета сырья (в качестве аналогов рассматриваются BI- и MES-системы) приводит к следующим эффектам [11]:

– сокращается время ввода и обработки данных, в среднем на 36%, в диапазоне от 0 до 90%;

– уменьшается объем незавершенного производства - в среднем – на 32%, диапазон сокращения – от 0 до 100%;

– снижаются затраты на традиционные (бумажные) коммуникации между подразделениями - в среднем, на 67% ;

Необходимость проведения исследования продиктована современными потребностями модернизации производства и возможностями предприятий лесопромышленного комплекса по внедрению инновационных решений. Проведение исследования соответствует технической политике по обеспечению научно-технического прогресса в лесопромышленном комплексе, заявленной в Положении о Государственном комитете Российской Федерации по лесной, целлюлозно-бумажной и деревообрабатывающей промышленности.

ГЛАВА 2. МЕТОДИКА ИЗМЕРЕНИЯ ОБЪЕМА ШТАБЕЛЯ КРУГЛОГО ЛЕСА

2.1 Анализ существующих методов измерения объемов круглого лесоматериала

Вопрос точного учета сырья и производимой продукции является одним из важнейших в условиях рыночных отношений и постоянной борьбы за минимизацию издержек производства.

Существует большое количество разнообразных методов измерения объема леса. Все они отличаются друг от друга как по физическим принципам, заложенным в их основу, так и по способам вычисления объема.

Особенностью круглых лесоматериалов является высокий уровень погрешностей измерений объема, что приводит к недостачам или излишкам при ревизиях остатков лесоматериалов на складах, колебаниям расхода древесины на единицу продукции при переработке.

Большинство методов измерения объема круглого леса были разработаны более 20 лет назад [12]:

– поштучное измерение диаметра в верхнем торце и вычисление объема по таблицам;
– поштучное измерение объема по серединному диаметру;
– геометрический групповой метод измерения;
– весовой групповой метод.

2.1.1 Поштучное измерение диаметра в верхнем торце

Поштучное (табличное) измерение круглых лесоматериалов имеет наиболее широкое распространение. Метод заключается в измерении диаметра верхнего торца и нахождении объема по таблицам объемов (ГОСТ 2708-75). По причине интенсивного использования в настоящее время вершинной части хлыстов средняя фактическая сбежистость сортиментов

оказалась выше значений, использованных при составлении этих таблиц. Поэтому фактическое значение объема большой партии сортиментов даже без учета припусков может значительно (до 10 %) отличаться от их объема по ГОСТ. Вместе с тем табличный метод весьма удобен для использования благодаря единообразию учета, относительной легкости реализации при измерениях вручную и удовлетворительной воспроизводимости результатов учета лесоматериалов на всех стадиях транспортировки и технологической переработки.

Измерение диаметра производится лесной вилкой или линейкой. Пороки и механические повреждения не должны оказывать влияния на результаты измерения диаметра. За средний диаметр принимают полусумму наибольшего и наименьшего диаметров в верхнем торце (без коры) [13]

2.1.2 Метод концевых сечений

Метод предусматривает измерение диаметра в нижнем и верхнем торцах и длины бревна. При наличии закомелистости диаметр измеряется на расстоянии 50см от нижнего торца. Объем бревна может быть определен по формуле:

$$V = \frac{\pi(d^2+D^2)}{8 \cdot 1000} L \quad (1)$$

где d и D – диаметр бревна в верхнем и нижнем торцах, см; L – длина бревна, м.

При поштучном определении объем лесоматериалов можно вычислять с помощью таблиц, официально принятых в стране поставки и согласованных с потребителем. Таблицы содержат объемы бревен для определенного интервала значений и диаметров, с учетом сбега хлыста. [13]

2.1.3 Геометрический метод измерения

Групповой геометрический метод учета круглых лесоматериалов основан на измерении линейных геометрических параметров штабеля

(длины, ширины, высоты), и вычислении плотного объема древесины с использованием переводного коэффициента.

Суть метода заключается в измерении объема каждого штабеля отдельно. Штабель может находиться в вагоне или автомобиле. Оценка среднего диаметра может производиться до погрузки или после разгрузки штабеля.

Длину, ширину и высоту штабеля измеряют мерной рейкой, штангой или рулеткой, длина которых должна превышать измеряемый размер штабеля. Результаты измерения округляют до 0,01 м.

Объем штабеля определяют по формуле

$$V_{\text{пл}} = V_{\text{скл}} k \quad (2)$$

где $V_{\text{пл}}$, $V_{\text{скл}}$ – объем плотного и складочного объема штабеля, м^3; k - коэффициент полнодревесности.

Геометрический метод не обеспечивает высокой точности измерения; кроме того, он трудоемок и опасен при выполнении работ в вагонах и на автомобильном транспорте. [13]

2.1.4 Измерение объема методом гидростатического взвешивания

Метод базируется на законе Архимеда. Установка гидростатического взвешивания состоит из весов, захвата, обеспечивающего обжим пакета бревен, устройства для полного погружения пакета в воду и погружения захвата до фиксированного постоянного уровня. Установка должна обеспечивать проведение измерений с погрешностью не более 1% и округлением результата до ближайших 0,05 т для следующих показателей: масса брутто (пакет, захват) в воздухе (m_1), т; масса тары (захвата) в воздухе ((m_2),), т; масса брутто (пакета бревен и захвата) после пофужения в воду ((m_3),), т; масса тары (захвата)' после пофужения в воду до фиксированного уровня (m_4), т.

Масса штабеля лесоматериалов равна $m = m_1 - m_2$

Объем лесоматериалов в штабеле $V_{\text{ш}}$ составляет:

$$V_{\text{ш}} = \frac{1}{\rho}(m + m_4 - m_3) \quad (3)$$

где р - плотность воды, р = 1,0 т/м^3.

Данный метод не нашел широкого применения из-за отсутствия надлежащих условий на сплаве – наличие изолированного бассейна для проведения измерений [14].

2.1.5 Весовой метод измерения

Измерение объема весовым методом проводят для штабелей лесоматериалов, составляющих вагонную, судовую или автомобильную партию. Массу лесоматериалов в партии определяют как разность между массой брутто и массой тары (вагоны, автомобиля).

Допускается измерение массы партии лесоматериалов, погруженных на судно по измерениям осадки судна, выполненным по нормативным документам на морском или речном транспорте. Объем лесоматериалов в партии определяют по формуле:

$$V = {}^{m}/_{k}, \quad (4)$$

где m – масса пачки лесоматериалов, определенная взвешиванием, т; k – коэффициент перевода массы лесоматериалов в м^3.

Значение k определяется экспериментально и зависит от месторасположения лесного массива, климатических условий, времени года, породного состава, сроков хранения лесоматериалов. В лесозаготовительной практике весовой метод не находит применения из-за длительного срока, в течение которого древесина от заготовителя поступает к потребителю. За это время влажность древесины, а следовательно, и масса сильно изменяются [14].

Для взвешивания лесоматериалов могут быть использованы устройства для измерения масс пачек на кранах; стационарные автомобильные и железнодорожные весы; весы автомобильные поосного взвешивания

(последние применяются для взвешивания автопоездов в движении и устанавливаются в дорожное полотно).

2.1.6 Бесконтактные методы измерения

Все методы, используемые в официальных ГОСТах и стандартах в нашей стране, осуществляют процесс измерения путем непосредственного взаимодействия с измеряемым объектом (контактные методы). К настоящему времени реализован ряд измерительных систем, процесс измерения в которых состоит в регистрации отраженного излучения (различной природы) от объекта измерения и глубокой компьютерной обработки результатов регистрации (бесконтактные методы). Это позволяет обеспечить методам следующие достоинства:

- отсутствие непосредственного контакта с объектом измерения;
- объективность измерения (отсутствие человеческого фактора);
- высокая точность и повторимость;
- отсутствие разницы в количестве измеряемых бревен;
- высокая производительность.

Можно выделили три основных направления исследований в этой области с позиции природы регистрируемого излучения. Оптические и ультразвуковые методы измерения объема в процессе измерения регистрируют излучение источником которого сами же и являются (Рисунок 2.1). Это несколько усложняет технологию и требует дополнительного оборудования. Измерение параметров объекта на основании изображения полученного с помощью фото или телекамер использует естественное освещение (Рисунок 2.2). И в этом отношении фотометрический метод наименее требователен к количеству и сложности оборудования. [12]

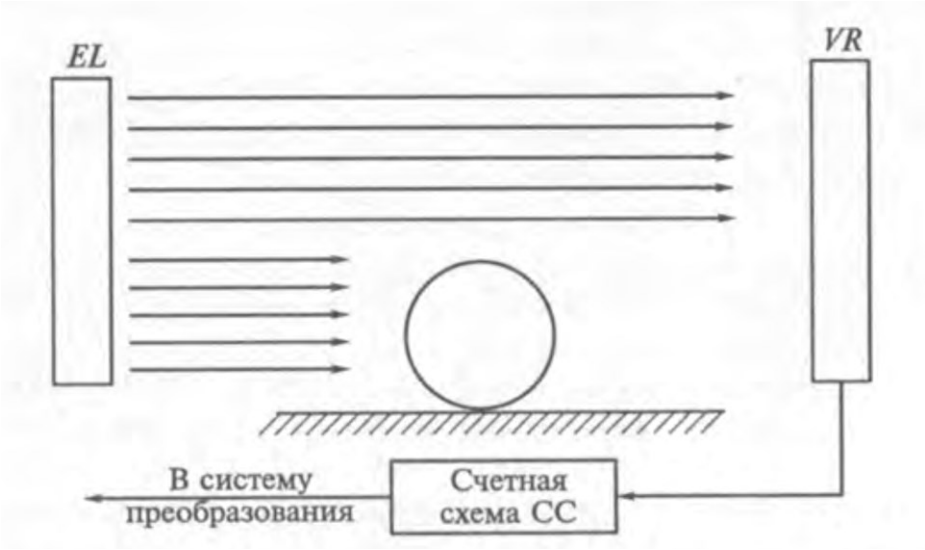


Рисунок 2.1 – Схема фотоэлектрического измерительного устройства

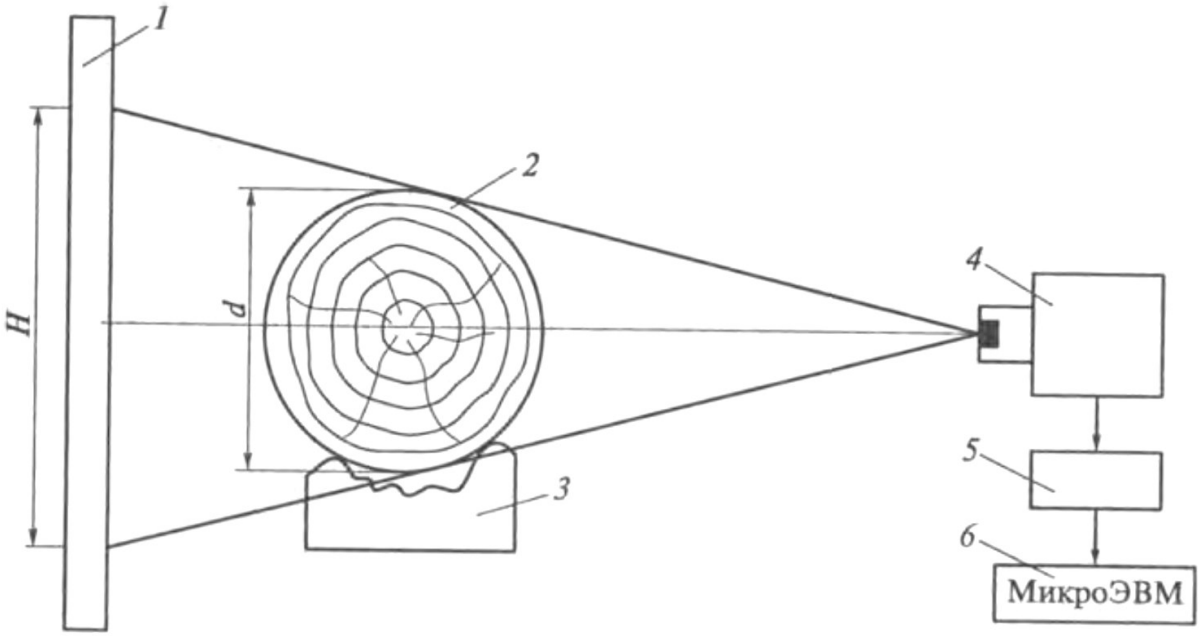


Рисунок 2.2 – Телевизионный измеритель диаметров. 1 – плоский осветитель, 2 – сортимент, 3- траверса, 4 – телевизионная камера, 5 – преобразователь аналогового сигнала, 6 – микроЭВМ

Автоматическое измерение объема круглых лесоматериалов заключается в определении фактического (физического) объема при помощи автокубатурников. При измерении истинного объема кубатурниками объем бревна рассматривают как сумму элементарных цилиндров, диаметр которых

изменяется по длине бревна в соответствии с его фактическим сбегом и особенностями формы. [13]

Основные требования, предъявляемые к автокубатурникам, по которым определяют объем бревен по его геометрическим размерам, сводятся к возможности измерения бревен любого диаметра и длин; измерению диаметра бревен без учета толщины коры; измерению длины бревна без учета припуска; возможности определения сбежистости бревна для корректирования объема сортимента; определении породы древесины и записи полученных результатов для дальнейшей их обработки и отчетности..

К кубатурникам относятся автоматы для штучного учета сортиментов, автоматы, определяющие истиный объем древесины и автоматы, определяющие объем сортиментов по таблицам автокубатурниками табличных объемов. [14]

При всех указанных достоинствах автокубатурников, необходимо отметить два критических недостатка:

– высокая стоимость,
– отсутствие мобильности.

2.1.7 Классификация методов измерения

Один из вариантов классификации методов измерения объема лесоматериалов по способу взаимодействия с объектом в процессе измерения предложен в [12]. Данная классификация приведена на Рисунке 2.3.

Рисунок 2.3 – Классификация методов измерения объема круглого леса по способу взаимодействия с объектом в процессе измерения

2.2 Описание предлагаемой методики

Методика определения количественных характеристик круглых лесоматериалов относится к области измерительной техники и может использоваться на различных стадиях заготовки и обработки лесоматериалов для технического контроля и учета заготовленной древесины. Методика заключается в фотографировании двух сторон штабеля бревен с дальнейшей оцифровкой и обработкой фотографий специализированным программным обеспечением. В основе метода лежит принцип цифровой фотограмметрии, который позволяет определять геометрические и количественные свойства объекта измерения по его изображениям. В ходе обработки изображении программными средствами производится автоматическое определение всех видимых торцов бревен каждого изображения, восстановление трехмерной пространственной структуры бревен, образующих штабель, и расчет его количественных характеристик. К количественным характеристикам, определяемым предложенным способом относятся: верхний и нижний диаметр и объем каждого бревна, а также объем всего штабеля. При отсутствии возможности фотографирования штабеля с двух сторон допускается производить съемку только одного торца, при этом определение трехмерной структуры и расчет некоторых параметров штабеля производятся при заданных ограничениях, исходя из принятой модели и известной априорной информации об объекте измерения.

Известен способ определения объема круглых лесоматериалов, описанный в заявке РФ №99109434, МПК G06N 33/46, опубликованной 10 февраля 2001 года на изобретение: «Фотометрический способ определения объема и качества круглых лесоматериалов», в котором для определения диаметра торцов бревен и расчета полезного объема осуществляют съемку торцевых сторон бревен. Для этого размещают, как минимум, одну управляемую стереопару, состоящую из видеокамер, ориентированных для съемки торцов, устанавливают маркеры с заданным расположением их в

пространстве таким образом, чтобы они попали в угол обзора обеих камер, регистрируют и получают изображения торцов в виде пары (стереопары) растровых изображений, поочередно вводят полученные изображения в компьютер, который преобразует их в эквивалентные цифровые изображения с соответствующей разрешающей способностью, размещают левое и правое изображения для получения стереоскопического эффекта, проводят компьютерную обработку изображений путем цветового выделения торцов, векторизации изображений (если съемка велась под углом), приводят в плоскость, параллельную плоскости экрана, восстановления недостающих границ торцов бревен и приведения всех торцов к единой плоскости, наблюдают объемную геометрическую модель бревен, приводят ее к плоскости маркеров, аппроксимируют каждый торец известными способами с минимальной погрешностью, получают теоретическую модель бревен, по которой с помощью ЭВМ определяют наименьший диаметр каждого торца и/или площадь, по известному заранее соотношению переходят от системы координат экрана к системе координат объекта и вычисляют объем исследуемых круглых лесоматериалов по известным методикам.

Описанное техническое решение имеет ряд недостатков, ограничивающих сферу его применения. Способ предполагает использование дорогостоящей измерительной техники (стереокамер) и установки маркеров с определенным расположением в пространстве для определения внешнего ориентирования камер и дальнейшего приведения плоскости сцены к плоскости маркеров для точного определения границ торцов, что усложняет измерение в реальных условиях промышленного предприятия и требует высокой квалификации обслуживающего персонала.

К наиболее близким по технической сущности к предлагаемому методу определения количественных характеристик круглых лесоматериалов, уложенных в штабель можно отнести «Способ измерения объема круглых лесоматериалов» (RU, 2362164 C2 МПК G01N 33/46, опубл. 20.07.2009), заключающийся в съемке торцов бревен, получении их изображений,

преобразований их в цифровой эквивалент и получении параметров теоретической модели бревен, основанный на том, что на стадии съемки на торцы некоторых бревен и стволы некоторых наружных бревен штабеля прикрепляют эталоны известной площади и длины в количестве, достаточном для получения требуемой точности, и занимающие часть соответствующего торца бревна и длины ствола бревна, осуществляется фотографирование обоих торцов и боковой поверхности штабеля, полученные изображения оцифровываются и обрабатываются с помощью компьютера для определения точных границ контуров торцов бревен и эталонов, определяют площадь торцов, эталонов и длины штабеля на фотографиях, вычисляют поверхность значений коэффициентов подобия на основании сравнения полученных после обработки снимков размеров эталонов и истинных их размеров, затем с учетом коэффициентов вычисляют площадь торцов и длину бревен, на основании которых по принятой модели определяется объем лесоматериалов.

Предложенный метод относительно эффективен для обеспечения точности результатов измерения при условии, что определено большое количество шаблонных объектов, но при этом требует большого количества ручных операций на стадии съемки, для нанесения эталонных объектов по периметру штабеля. К тому же предложенный способ, по всей видимости, не позволяет определять геометрические и количественные характеристики каждого бревна.

Суть предлагаемого метода заключается в получении изображений двух торцевых сторон штабеля бревен с дальнейшей обработкой фотографий специализированным программным обеспечением. В ходе обработки изображении программными средствами производится:

– автоматическое определение всех видимых торцов бревен каждого изображения;

– восстановление трехмерной пространственной структуры штабеля;

– расчет количественных характеристик каждого бревна и штабеля в целом.

К названным количественным характеристикам относятся: верхний и нижний диаметр и объем каждого бревна, а также объем всего штабеля. При отсутствии возможности фотографирования штабеля с двух сторон допускается производить съемку только одного торца, при этом определение трехмерной структуры и расчет некоторых параметров штабеля производятся при заданных ограничениях, исходя из принятой модели и известной априорной информации об объекте измерения.

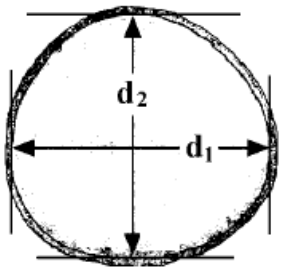
Поскольку качество производимых измерений существенно зависит от расположения съемочных датчиков, при проведении съемок штабеля необходимо располагать фотокамеру параллельно плоскостям, образуемым торцами бревен. В таком случае изображения объекта измерения выглядят как копии этих плоскостей в уменьшенном масштабе. Следовательно, при таком взаимном расположении объекта измерения и камеры можно определять параметры бревен непосредственно в системе координат снимка, считая изображение масштабированной копией пространственного объекта. Такая модель съемки позволяет относительно просто рассчитывать масштабные коэффициенты, определяющие переход от пикселей изображения к единицам измерения штабеля – миллиметрам. Для проведения процедуры калибровки, используется метод сопоставления с эталоном. Для этого осуществляется распознавание одного или нескольких шаблонных объектов на изображении, физические размеры которых должны быть заранее измерены и введены в компьютер до программной обработки результатов. Коэффициенты подобия рассчитываются как отношение реальных и пиксельных размерам эталонов. Предложенным способом допускается использовать в качестве эталонного объекта одно или несколько бревен измеряемого штабеля. Единственное требование, предъявляемое к

калибровочным объектам: они должны располагаться в плоскости срезов бревен и быть доступными для измерения программными средствами.

В ходе обработки изображений программными средствами интересующие нас объекты (срезы бревен) выделяются автоматически по признакам формы и пространственного расположения; после чего для выделенных объектов проводится уточнение границ их контуров на основании разности цветовых характеристик торцов бревен и фона сцены. Для каждого контура находится эквивалентный эллипс, т.е. эллипс, который наиболее точно аппроксимирует данный контур, и в дальнейшем под площадью торца бревна будет пониматься площадь эквивалентного эллипса. Это делается с целью соответствия метода измерения диаметров бревен требованиям ГОСТ 32594-2013 «Лесоматериалы круглые. Методы измерений», где для бревен, которые по визуальной оценке признаются овальными, проводят два измерения диаметра, одно перпендикулярно другому, и вычисляют их среднеарифметическое значение. В случае замены контура торца эквивалентным эллипсом, двум измерениям диаметра соответствует большая и малая оси этого эллипса, а диаметр торца бревна рассчитывается из выражения:

$$d = \frac{d_1 + d_2}{2} \tag{5}$$

где d_1 и d_2 – большая и малая оси эквивалентного эллипса



После того, как в результате проведенных операций распознавания и калибровки становятся известны геометрические характеристики, определяется ориентация бревен в штабеле путем реконструкции пространственной модели штабеля.

Для расчета объема каждого бревна используются

– расчетные значения диаметров торцов бревен (или торца в случае одного изображения),

– значение длины штабеля (задается фиксировано для всего штабеля);

– значение поправочного коэффициента на объем коры (задается фиксировано для всего штабеля);

– значение коэффициента сбежистости (задается фиксировано для всего штабеля);

– калибровочные коэффициенты.

На заключительном этапе рассчитывается объем штабеля круглого леса путем складывания объемов отдельных бревен в штабеле:

$$V = \sum_{i=0}^{n} V_n \quad (6)$$

где n – количество бревен в штабеле, V_n – объем n-го бревна

2.3 Выводы

Практически все контактные методы имеют ошибку измерения, достигающую существенных величин. Более того, она превосходит установленные отечественные стандарты. С другой стороны все эти методы имеют большую трудоемкость и слабо поддаются автоматизации. Человеческий фактор играет большую роль в технологии измерения контактными методами, а он может привести к погрешностям сколь угодно больших размеров.

Даже если удастся разработать метод, который будет удовлетворять отечественным стандартам по точности измерения, его трудоемкость окажется столь высокой, что его использование будет нерентабельно. К тому же остается проблема автоматизации процесса измерения. [12]. Применение бесконтактных автокубатурников позволяет решить данные задачи, но остается проблема мобильности метода учета.

В предложенной методике измерения объема штабеля бревен технический результат достигается за счет упрощения процесса формирования первичной информации об объекте измерения вплоть до выполнения расчетов по изображению одного торца штабеля бревен, что является критическим условием – зачастую получение изображения второго торца невозможно или сильно затруднено. Низкий процент ошибок первого и второго рода при поиске и выделении торцов бревен на изображении достигается благодаря использованию признаков цвета, формы и пространственной ориентации объектов интереса в пределах изображения. Точность расчетов повышается за счет реконструкции трехмерной модели каждого бревна в штабеле, определяющей действительное положение, размеры и ориентацию (комель/вершина) бревна через сопоставление двух изображений штабеля, использования методов расчета исходя из существующих требований к поштучному измерению объема круглых лесоматериалов.

В следующих разделах приведено подробное описание отдельных этапов работы алгоритма автоматического измерения штабеля бревен по предложенной методике.

ГЛАВА 3. РАЗРАБОТКА И ИССЛЕДОВАНИЕ МЕТОДОВ ЦИФРОВОЙ ОБРАБОТКИ И АНАЛИЗА ИЗОБРАЖЕНИЙ ДЛЯ СОЗДАНИЯ АЛГОРИТМА АВТОМАТИЧЕСКОГО ВЫДЕЛЕНИЯ ТОРЦОВ БРЕВЕН НА ИЗОБРАЖЕНИИ

Развитие вычислительной техники и средств получения цифровых изображений в последнее десятилетие обеспечило возможность применять методы машинного зрения для решения задач бесконтактных измерений и создания трехмерных компьютерных моделей объектов реального мира. Совершенствование средств получения цифровых изображений и методов их анализа обусловили необходимые предпосылки для появления высокопроизводительных бесконтактных измерительных систем, применяемых в прикладных инженерных задачах, обеспечив тем самым высокую точность и высокую степень автоматизации измерений в промышленности и на производстве. Определение объема круглых лесоматериалов является весьма востребованной задачей в области технического контроля в лесодобывающих и лесоперерабатывающих областях промышленности.

При создании программного комплекса решался широкий класс теоретических и прикладных задач, относящихся к области машинного зрения, а также смежных дисциплин, таких как цифровая фотограмметрия и теория распознавания образов. Основными задачами, решаемыми при создании программы по расчету объема бревен, уложенных в пакет, являлись следующие:

– выбор конфигурации съемки,
– калибровка сенсоров,
– разработка алгоритма автоматического выделения округлых объектов на изображении,
– построение модели штабеля бревен,
– разработка алгоритма определения объема круглых лесоматериалов,

Таким образом, настоящая работа разделена на несколько частей, в которых проводится аналитический обзор существующих разработок, излагаются теоретические положения, описываются технические методы для решения задачи мобильного измерения штабелей бревен на основе фотограмметрии.

3.1 Обзор научной литературы по методам выделения круглых объектов на изображении

Проблема автоматического детектирования наблюдаемых объектов на статической или динамической сцене на протяжении долгого времени остается одной из актуальных задач в области компьютерных наук. В ходе работы проанализировано большое количество научных трудов, касающихся выделения круглых объектов на изображении и других задач, близких по содеожанию.

В работе [15] предлагается строить множество базовых признаков над картиной направленных краев (детектор Канни); используется модернизированный алгоритм Виолы-Джонса с суженным пространством признаков Хаара. Полученный алгоритм устойчив к различным условиям освещенности и шуму и не дает ложных срабатываний, при этом полнота составляет 87%. Метод Виолы-Джонса используется также в задаче распознавания лиц [16], но высокая точность – 98% – достигается за счет сравнения признаков лица на входном изображении, полученных в результате вейвлет-преобразования, с эталонами в базе данных, т.е. специфика задачи не соответствует предметной области. Другой подход к решению задачи выделения лиц – комбинация операторов симметрии, методов поиска по шаблону и нейронных сетей [17] – демонстрирует большие вычислительные затраты.

Алгоритм выделения и обнаружения воздушных объектов, основанный на оценивании параметров АРСС-модели фона с помощью процедуры адаптивной пространственной фильтрации [18], суть которого в

классификации точек исходного изображения на точки, принадлежащие объекту, и точки, принадлежащие фону, показывает недостаточную эффективность при большой степени зашумленности и малом контрасте объекта. Пирамидальный алгоритм сегментации изображений [19] на основе использования цветовых и текстурных различий областей предполагает ручной подбор значений коэффициентов и порогов, при этом высока вероятность ошибки сегментации одноцветных объектов. Критерием однородности является оценка близости элементов и областей изображения в объединенном текстурно-цветовом пространстве признаков, но эксперименты с различными цветовыми пространствами показали невозможность однозначного отделения целевых объектов от фона на основе цветового признака. В методе отыскания круглых форм на основе техники роевого интеллекта [20,21] используется целевая функция для сравнения подобия круга кандидата с фактическим кругом на граничной карте входного изображения, основанного на различии местоположений их центров и длин радиусов. Основным недостаткам метода является необходимость знать количество искомых фигур заранее. В ряде работ рассмотрено использование классификаторов и решающих деревьев – классификация фигур, вписанных в прямоугольник [22], например детекторы образов колес в системе автоматической классификации транспортных средств на основе классификаторов Виолы-Джонса [23] Интересный подход представлен в работе по распознаванию зданий на снимках из космоса [24]: участки кривых на изображении обнаруживаются с помощью метода опорных векторов, после чего вычисляется нормализованный разностный вегетационный индекс (NDVI) и нормированная цифровая модель поверхности (NDSM), далее границы выявляются с помощью метода Хафа, и, наконец, выявленные линии группируются, образуя конечную картину. Расчетная точность достигает 98,22%, но для этого требуется высокое разрешение исходного изображения. Часть методов основана на поиске и группировке дуг. Так, в работе [25] описан метод прослеживания в видеопотоке объектов,

содержащих множество концентрических дуг, основанный на применении структурного тензора и голосующей схемы в пространстве центров дуг. Другой метод предлагает схему обнаружения эллипсов с использованием сегментов кривых [26] – попарно проверяет их принадлежность к одному предполагаемому эллипсу и, если произошло совпадение, кривые объединяются в один сегмент. Метод позволяет детектировать эллипсы в условиях плохой освещенности, шума, перекрытия изображений, но при этом часты случаи ложного детектирования. Еще один иерархический подход – объединение отрезков, потенциально принадлежащих к одному эллипсу, по условиям связности и кривизны, группировка дуговых сегментов, принадлежащих к одному эллипсу и статический метод RANSAC для определения соответствий между дугами и эллипсом [27]. К недостаткам относится низкая эффективность для объектов малых размеров. Большая группа работ использует в своей основе метод Хафа [28-30]. Метод на основе свойств изофот [28] предполагает выбор наиболее значимых краевых пикселей и их классификацию в подмножества кривизны равной изофоте. Анализ предполагаемых кругов выполняется стратегией голосования на основании оценки плотности ядра, за которым следует уточнение алгоритма, основанное на линейной компенсации ошибок. Метод обнаружения круглых объектов посредством векторов градиента [29] – алгоритм для нахождения круглых объектов, которые ярче или темнее фона. Сначала авторы вычисляют градиентные вектора круглых объектов на изображении, следующим шагом выполняется поиск противоположно направленной пары векторов, лежащих на противоположных концах окружности, и на финальном этапе выделяются все круги из кандидатов на предыдущем шаге. Метод нахождения круглых объектов с помощью симметрии [30] основан на геометрических свойствах: краевые точки во входном снимке разделяются на несколько изображений, причем эллипсы с различными симметричными осями лежат в разных изображениях, и в каждом изображении применяется

симметрия до получения пяти параметров одного эллипса (координаты центральной точки, угол для ориентации, большая и малая оси эллипса).

Распознаванию непосредственно срезов бревен на изображениях было посвящено несколько работ [31-39], некоторые из них нашли практическое применение в составе прикладных измерительных систем [38-39]. Предложенные в данных работах методы детектирования можно разделить на две категории. К первой группе относятся методы, основанные на машинном обучении. В [36] авторы предложили итеративную схему детектирования и сегментации, в которой на этапе обнаружения срезов бревен используются дескрипторы особых точек на основе гистограммы направленных градиентов (англ. HOG – histogram of oriented gradients) [32] совместно с признаками Хаара и операторами локальных двоичных шаблонов (англ. LBP – local binary patterns) [43]. В работе [35] для поиска бревен по их ключевым признакам используется метод Виолы–Джонса[44]. Основная его идея заключается в использовании каскада классификаторов, каждый из которых является комитетом (ансамблем) слабых классификаторов. В качестве признаков для алгоритма распознавания используются прямоугольные признаки Хаара.

Методы второй группы построены по схеме обучения без учителя и используют предположения об известной форме и размерах бревен [34,38] В основном эти методы основаны на преобразовании Хафа [45-49] или его модификациях и применяются для обнаружения срезов бревен на изображении в виде окружностей или эллипсов. Часть из них не соответствует специфике задачи ввиду высокой вычислительной сложности заявленных алгоритмов [50-53], низкой полноты распознавания [54,55] или чувствительностью к шуму и другим видам искажений [56-57], но в целом методы данной группы показывают высокую эффективность распознавания целевых объектов.

3.2 Разработка алгоритма выделения торцов бревен на изображении

За последнее десятилетие создано множество успешных систем машинного зрения, имеющих различное применение, при этом характерно, что последовательность обработки изображений в них приблизительно одинакова и сводится к определенной последовательности действий. Такая закономерность, на наш взгляд, связана с применением авторами этих систем, так называемой модульной парадигмы, предложенной Д. Марром [58]. Эта парадигма утверждает, что обработка изображений опирается на несколько последовательных уровней: начиная с низшего, связанного с пиксельной обработкой (фильтрация шумов, гистограммная обработка) и заканчивая верхним уровнем, связанным собственно с пониманием изображений. Исходя из этого, в области машинного зрения принято выделять следующие основные этапы обработки данных:

- предварительная обработка изображений
- сегментация
- выделение геометрической структуры
- определение относительной структуры и семантики

Применяя данный принцип модульности к разработке алгоритма определения объёма круглых лесоматериалов по серии изображений, решение задачи сводится к реализации следующей последовательности:

- детектирование объектов
- выделение и классификация интересующих нас объектов
- сегментация
- определение геометрических характеристик
- восстановление трехмерной структуры
- измерение

3.2.1 Детектирование объектов

Этап детектирования заключается в выделении на изображении как можно большего количества интересующих нас объектов по ряду признаков. Цель детектирования бревен заключается в получении первичных входных данных об объектах измерения, выделении необходимых групп признаков для дальнейшей их классификации и сегментации. Анализ большого числа изображений с бревнами показывает, что большинство бревен имеют круглую или близкую к ней форму (Рисунок 3.1).

а

б

в

г

д

е

Рисунок 3.1 – Примеры исходных изображений

Анализ существующих методов показал, что одним из эффективных методов обнаружения окружностей на изображении является преобразование

Хафа и смежные с ним группы методов. К недостаткам указанных подходов следует отнести следующие:

– преобразование работает с бинарным изображением;
– преобразование Хафа имеет большую вычислительную сложность;
– для хранения массивов данных требуется большой объем памяти;
– алгоритмы сильная чувствительны к искажениям и шумам.

Наиболее предпочтительным с точки зрения вычислительной трудоемкости и требований к минимальным искажениям изображений объектов является метод, предложенный в работе [59], основанный на вычислении быстрого симметричного преобразования. Суть метода заключается в следующем.

Для каждого радиуса r рассчитывается массив ориентаций O_r и амплитуд M_r путем анализа градиента g в каждой точке p, для которой определяются координаты пикселей, соответствующих положительно и отрицательно направленным точкам $p_{+ve}(p)$ и $p_{-ve}(p)$ (Рисунок 3.2) по следующим формулам:

$$\begin{aligned} p_{+ve}(p) &= p + round\left(\frac{g(p)}{\|g(p)\|}n\right) \\ p_{-ve}(p) &= p - round\left(\frac{g(p)}{\|g(p)\|}n\right) \end{aligned} \tag{7}$$

В массивах ориентаций и амплитуд точка p для каждой $p_{+ve}(p)$ увеличивается на 1 и $\|g(p)\|$ соответственно, тогда как для $p_{-ve}(p)$ значения уменьшаются на соответствующие величины. Таким образом,

$$\begin{aligned} O_n\big(p_{+ve}(p)\big) &= O_n\big(p_{+ve}(p)\big) + 1, \\ O_n\big(p_{-ve}(p)\big) &= O_n\big(p_{-ve}(p)\big) - 1, \\ M_n\big(p_{+ve}(p)\big) &= M_n\big(p_{+ve}(p)\big) + \|g(p)\|, \\ M_n\big(p_{-ve}(p)\big) &= M_n\big(p_{-ve}(p)\big) - \|g(p)\|. \end{aligned} \tag{8}$$

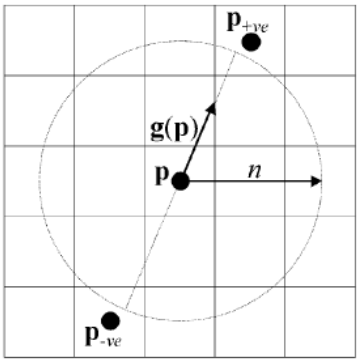


Рисунок 3.2 – Определение положительно и отрицательно направленных точек

Массив F_r для данного радиуса r определяется как произведение массива амплитуд M_r и ориентаций O_r:

$$F_r = M_r \cdot O_r^{\alpha} \quad (9)$$

где α – степень жесткости детектора, показывающая, насколько точно наблюдаемые объекты должны соответствовать форме круга.

Как правило, в том числе и в имеющейся задаче детектирования торцов бревен, направление градиента не имеет значения, в этом случае массив рассчитывается как

$$F_r = sgn(M_r) \cdot O_r^{\alpha} \quad (10)$$

Полное преобразование определяется как сумма сглаженных массивов для всех рассматриваемых радиусов r:

$$S = \sum_{r\,=\,r_{min}}^{r_{max}} S_r \quad (11)$$

где S_r определяется как свертка массива F_r с ядром Гаусса A_r:

$$S_r = F_r * A_r \quad (12)$$

Данный метод быстро и качественно работает на изображениях с априорно известными радиусами, допускающих небольшие искажения формы, и при условии, что разброс радиусов поисков лежит в небольшом диапазоне. Последние ограничения не позволяют использовать этот метод в явном виде к задаче детектирования срезов бревен. Из специфики данной задачи следует выделить следующие аспекты, влияние которых нельзя обойти при разработке алгоритма детектирования:

– отсутствует априорная информация о радиусах объектов на изображении;

– срезы бревен имеют не строго округлую форму, а близкую к ней (особенно комлевые стволы);

– для срезов некоторых пород древесины характерно наличие концентрических окружностей (годовалые кольца или сердцевины);

– из-за неравномерной укладки бревен в штабели, нередки случаи загораживания торцов соседними бревнами.

Исходя из сказанного выше, для решения задачи детектирования бревен был предложен модифицированный метод поиска радиально симметричных объектов на изображении с учетом изложенных выше ограничений. Основные особенности работы нового преобразования заключаются в следующем.

1. Для задания диапазона радиусов поиска используется следующая гипотеза. Количество бревен в штабеле составляет не менее 10 (в противном случае можно выполнить ручное выделение малой партии бревен) и не более 200; эквивалентная площадь, занимаемая срезами бревен, может варьироваться в зависимости от изображения, но среднее значение составляет примерно 25 процентов от площади изображения. При таких ограничениях диапазон радиусов поиска рассчитываться из следующих соотношений:

$$R_{min} = \sqrt{\frac{A \cdot p}{\pi \cdot N_{max}}}$$
$$R_{max} = \sqrt{\frac{A \cdot p}{\pi \cdot N_{min}}} \quad (13)$$

где A – площадь изображения в пикселях, p – процент площади, занимаемой штабелем от площади всего изображения, N_{max}, N_{min} – максимальное и минимальное количество бревен в штабеле.

Такой подход позволяет обработать большой диапазон наиболее вероятных размеров бревен. Большинство фотографий, на которых

анализировался алгоритм, соответствуют данной гипотезе, такие же требования указаны в руководстве пользователя программным продуктом, как рекомендации для наиболее качественной работы алгоритма.

2. Для снижения количества вычислений в модифицируемом алгоритме производится уменьшение исходного изображения до оптимальных размеров с использованием бикубической аппроксимации. Под оптимальным размером подразумевается такой размер, при работе с которым алгоритм обеспечивает должное качество детектирования объектов за разумное время с минимальной потребляемой памятью. С большой вероятностью окружности, найденные на уменьшенном изображении, будут соответствовать окружностям на оригинальной картинке. На основе анализа работы алгоритма детектирования при различных масштабах было установлено, что существует такое пороговое значение T части точек исходного изображения, которые участвуют в преобразовании, для которого количество вычислений существенно снижается без ухудшения качества детектирования. В ходе экспериментов с изображениями, полученными с мобильного устройства Samsung Galaxy Tab 3 GT0P5210 16 Gb 10,1” и имеющими разрешение 2048*1536 пикселей, было установлено оптимальное значение масштаба исходного изображения – 25%, что соответствует изображению с размером 1024*768 пикселей (Рисунок 3.3).

Из недостатков метода выделения ключевых точек на основе быстрой радиальной симметрии следует отметить:

– при большом диапазоне радиусов поиска отношение сигнал/шум суммарного массива существенно снижается, что осложняет поиск потенциальных центров окружностей. Это связано с тем, что лишь малая часть целевых объектов содержится в детектируемой выборке радиусов;

– есть ограничения, связанные с подбором коэффициента α. Для α, отличного от 1, вычислительная сложность алгоритма возрастает существенно, а при равенстве его 1 появляется большое количество ложно детектированных шумовых компонент. Для задачи детектирования бревен,

где преобладают существенные отклонения срезов бревен от формы круга, лучшим решением будет не задание коэффициента α через параметры алгоритма в явном виде, а определение радиальных искажений посредством сравнительного анализа выходных данных алгоритма для отбора лучших кандидатов.

– после обнаружения потенциальных центров окружностей в суммарном массиве S для уточнения радиусов нужно повторно обращаться к конкретным массивам S_r для сравнения откликов в найденных точках. Для большого диапазона радиусов поиска это требование накладывает ограничения по используемой памяти.

– Нельзя не учитывать проблему выбора оптимального порога при анализе пиков суммарного массива для конкретного изображения.

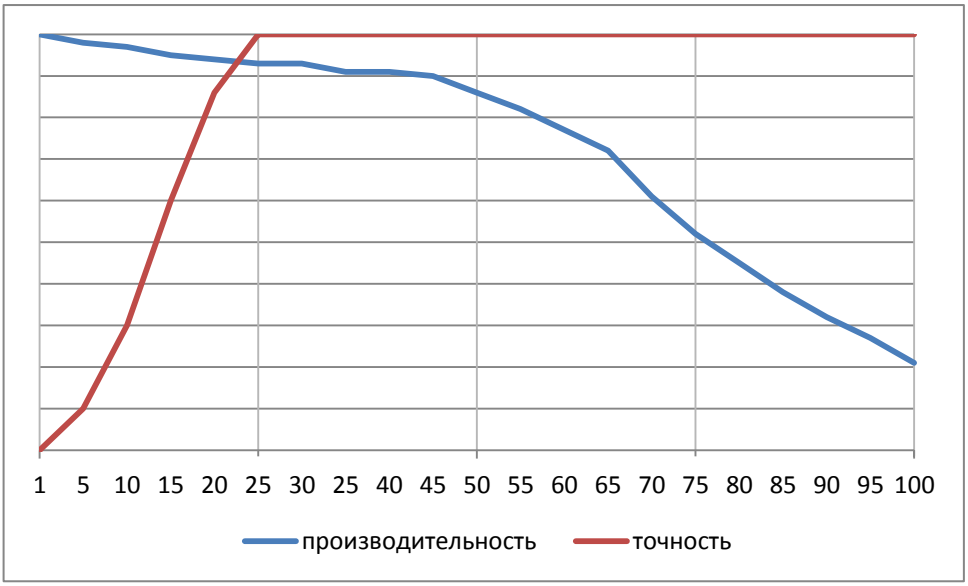
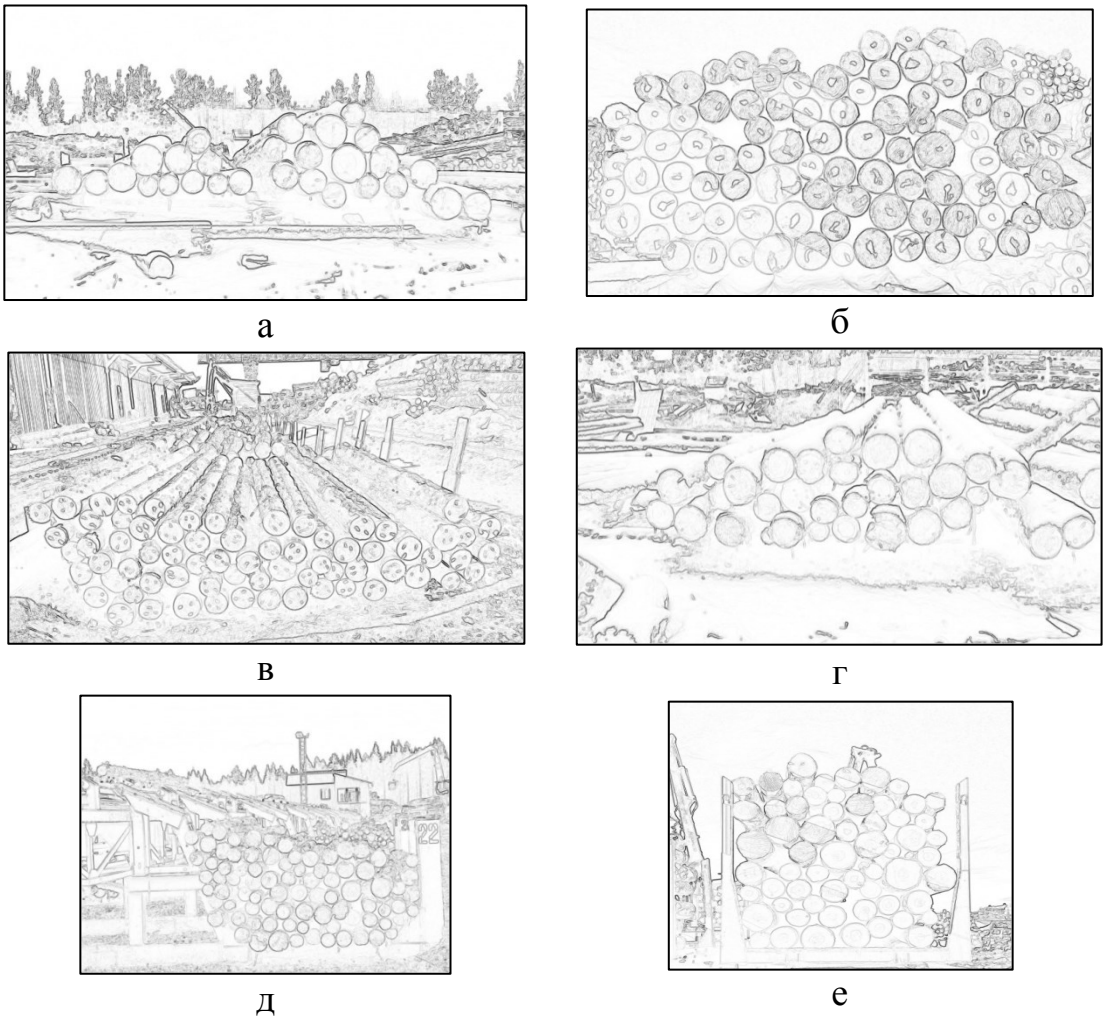


Рисунок 3.3 – Зависимость производительности и точности от масштаба изображения

Для поиска радиусов в модифицируемом алгоритме не строится массив амплитуд, а решения детектора принимаются на основе анализа модифицируемого массива ориентаций. Первым шагом алгоритма является процесс обнаружения пикселей края оператором Собеля, дающим оценку амплитуды и направления вектора градиента. Голосующими контурными точками являются точки с высоким значением модуля градиента. Для

каждого краевого пикселя производится оценка направления вектора градиента с целью оценки центра окружности радиуса R.



а б в г д е

Рисунок 3.4 – Применение оператора Собеля к исходным изображениям

На втором этапе анализа производится поиск локальных максимумов в массиве ориентаций для детектирования геометрических примитивов с определенными радиусами. Обычно для анализа пространства параметров применяется метод пороговой сегментации массивной функции. Однако такой подход не позволяет определить оптимальный порог, значение которого зависит от конкретного изображения. К тому же при анализе массивов целесообразно, чтобы точки локальных максимумов были инвариантны к изменению масштаба и учитывали искажения формы объектов. Рассмотрим предложенный способ анализа массивной функции для детектирования срезов бревен.

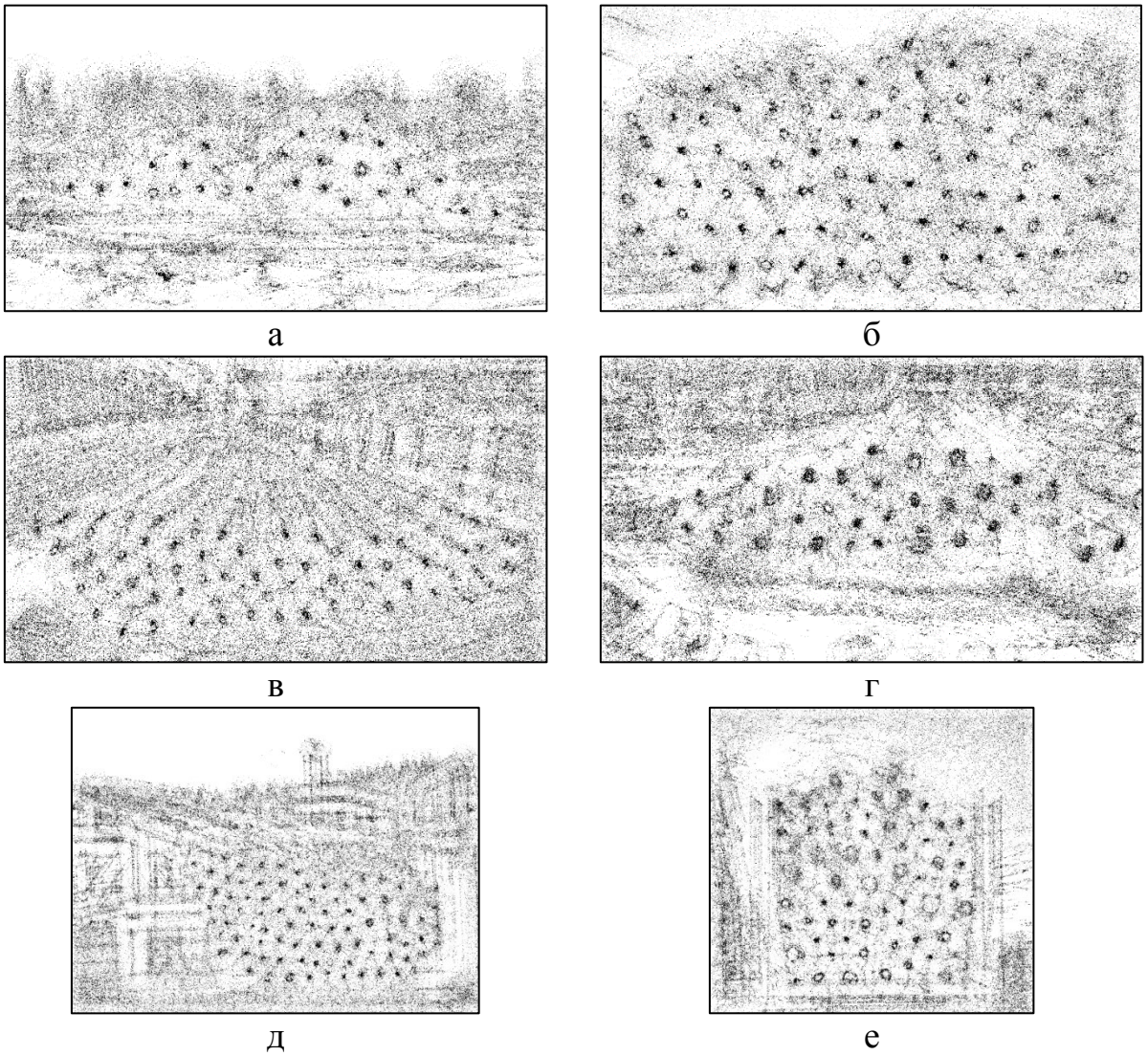


Рисунок 3.5 – Примеры массивов ориентаций для исходных изображений

Для учета инвариантности к масштабу и влияния искажений формы объектов алгоритм разбивает все множество радиусов поиска на неперекрывающиеся диапазоны. Каждый диапазон покрывает определенный интервал масштабов и имеет свой характерный размер фильтра. Размер фильтра учитывает допустимые искажения формы объекта и его размер. Очевидно, чем больше радиус объекта, тем более весомый вклад в локальный максимум массива будет сделан его краевыми точками, с другой стороны с увеличением радиуса целевого объекта или степени искажения относительно круглой формы увеличивается разброс интенсивности в массиве относительно истинного положения локального максимума. Таким образом, массив большего радиуса должен сканироваться окном большего размера.

Поэтому характерный размер апертуры окна сканирования рассчитывался как функция от радиуса поиска по следующей формуле.

$$A_r = \begin{cases} [\log_{10} Sq_r] \text{ для нечетных} \\ [\log_{10} Sq_r] + 1 \quad \text{иначе} \end{cases} \tag{14}$$

где [] – целая часть числа, Sq_r – площадь круга с радиусом r.

Основная идея метода определения оптимального порога заключается в том, что он может быть оценен из результатов работы алгоритма над изображением такового же размера как исходное, имеющего такое же яркостное распределение как у исходного, но не содержащего в себе объектов, обладающих радиальной симметрией. Иными словами, к изображению был применен алгоритм случайной перестановки его строк и столбцов, что позволило устранить информацию о пространственном положении круглых объектов, но сохранить сведения об интенсивности и ориентации каждого пикселя исходного изображения. Идея данного решения состоит в предположении о том, что величина оптимального порога зависит от энтропии исходного изображения. Тогда оптимальный порог для сегментации массива исходного изображения может быть выбран как

$$T_r = \max(\tilde{O}_r) + A_r^2 \tag{15}$$

$\max(O_r)$ – значение глобального максимума массива «шумового» изображения при сканировании окном с апертурой A_r.

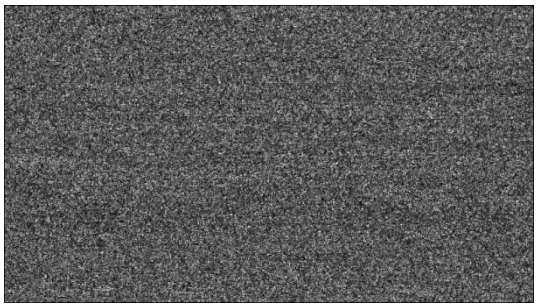


Рисунок 3.6 – Визуализация «шумового» изображения с энтропией изображения 3.2а

Величина A_r в выражении может трактоваться как поправочный масштабный коэффициент. А сам дескриптор интересующей нас области,

инвариантный к масштабу, для некоторого радиуса r рассчитывать следующим образом:

$$D_r(x,y) = O_r(x,y) - T_r \quad (16)$$

где $O_r(x,y)$ – значение локального максимума в массиве радиуса r с координатами (x,y) при сканировании окном апертурой A_r, T_r – оптимальный порог для радиуса r.

Таким образом, предложенный метод решает две задачи – поиск нужных объектов, обладающих свойством радиальной симметрии и создание их дескрипторов, инвариантных к масштабу. Это значит, что значение ключевой точки для массива радиуса r_1 будет такое же, как значение локального максимума в массиве r_2 ($r_1 \neq r_2$). А сама величина дескриптора (вес объекта) тем больше, чем точнее объекты интереса будут соответствовать форме круга. Такое решение позволяет производить сравнение и выбор лучших кандидатов среди найденных объектов. Результат работы модифицированного алгоритма детектирования объектов приведен на Рисунке 3.7.

Поскольку взаимная корреляция соседних радиусов поиска негативно влияет на результат работы (на изображении много пересекающихся окружностей), в алгоритм введена функция фильтрации, учитывающая взаимное перекрытие бревен. Также срезы бревен по своей природе могут иметь ярко выраженную сердцевину, что негативно влияет на результат работы детектора. Для учета этого необходимо ввести функцию, которая отдает предпочтение большему (внешнему) радиусу при условии, что окружности (бревно и его сердцевина) являются концентрическими.

а

б

в

г

д

е

Рисунок 3.7 – Результат работы алгоритма детектирования круглых объектов.

Фильтрация выполняется по следующему алгоритму:

1. К результирующему преобразованию модифицированного алгоритма детектирования применяется mean shift кластеризация [60], суть которой заключается в том, что для множества дескрипторов $\{d_i\}$ функция плотности распределения задается как

$$f(d) = \frac{1}{nh^2}\sum_{i=1}^{n} K\left(\frac{d-d_i}{h}\right), \quad (17)$$

где h – апертура окна, равная заданному минимальному радиусу бревна R_{min},

$K(d)$ – ядро. В данном случае используется ядро Епанечникова [61], обладающего радиальной симметрией

$$K(d) = \begin{cases} 1 - d, & 0 < d < 1 \\ 0, & d > 1 \end{cases} \quad (18)$$

Основной целью алгоритма среднего сдвига является перемещение точек в направлении локального увеличения плотности. Для оценки этого перемещения к функции плотности, рассмотренной выше, применяется градиент $\nabla f(d)$:

$$\nabla f(d) = \frac{2}{nh^4} \sum_{i=1}^{n} g\left(\left\| \frac{d-d_i}{h} \right\|^2 \right) \cdot m(d), \quad (19)$$

где $g(\|d\|^2) = -K'(\|d\|^2)$,

$m(d)$ – вектор среднего сдвига. Рассчитывается как

$$m(d) = \left(\frac{\sum_{i=1}^{n} d_i g\left(\left\| \frac{d-d_i}{h} \right\|^2 \right)}{\sum_{i=1}^{n} g\left(\left\| \frac{d-d_i}{h} \right\|^2 \right)} \right) - d \quad (20)$$

Вектор $m(d)$ всегда направлен в сторону максимального увеличения плотности. Алгоритм кластеризации дескрипторов $\{d_i\}$ имеет следующий вид:

а) Расчет вектора среднего сдвига для каждого дескриптора d_i

б) Сдвиг на $d_i \rightarrow d_i + m(d_i)$

в) Новая итерация до достижения устойчивого равновесия (центра масс, $m(d_i) \rightarrow 0$).

г) Дескрипторы с общим центром масс образуют кластер.

2. Для каждого кластера рассчитывается средневзвешенный радиус по формуле:

$$r_{\text{ср}} = \frac{\sum_{i=1}^{n} r_i * \omega_i}{\sum \omega_i}, \quad (21)$$

n – количество элементов в кластере, ω_i – весовая функция:

$$\omega_i = l_{r_i} * n_{r_i}, \quad (22)$$

где l_{r_i} – длина радиуса, n_{r_i} – количество радиусов данной длины в кластере.

Результат работы после фильтрации приведен на Рисунке 3.8.

а

б

в

г

д

е

Рисунок 3.8 – Результат алгоритма детектирования после фильтрации

3.2.2 Кластеризация

Кластеризация решает задачу объединения множества объектов с радиальной симметрией, полученных на этапе выделения, в группы, так чтобы получившиеся подмножества не пересекались и из этой структуры явно выделялись целевые объекты. Для определения объема бревен, уложенных в штабель, ставится задача разделения найденных объектов на два подмножества: кластер «штабель бревен» и кластер «объекты, не относящиеся к штабелю». Решение такой задачи позволит исключить из последующего процесса сегментации и измерения области, заведомо не относящиеся к интересующим нас объектам, но попавшие в совокупную выборку на предыдущем этапе.

Чтобы соединять или разбивать объекты в классы, нужно построить признаковое описание и определиться с метрикой, исходя из целей кластеризации. Цель кластеризации может быть сформулирована следующим образом: необходимо оставить бревна, лежащие в штабеле, и убрать как шум все остальные. Решение такой задачи требует выбора и введения определенных ограничений, зависящих от характера входной информации. Для заданной предметной области была введена геометрическая мера схожести (мера кучности бревен), определяющая, образуют ли бревна штабель. Среди существующих методов кластеризации [62]: статистические, иерархические, графовые предпочтение было отдано последней группе, из-за их наглядности и относительной простоты реализации (см. Рисунок 3.7).

К множеству выделенных объектов применен следующий алгоритм:

1. Триангуляция Делоне [63] итеративным методом с динамическим кэшированием [64];
2. Нахождение минимального дерева графа по алгоритму Борувки [65];
3. Разрез дерева по ребрам до тех пор, пока не будет достигнуто условие

$$\max_{(v,u)\in E} c(v,u) < 2 \cdot \max(r_{\text{ср}}), \tag{23}$$

где E – множество ребер графа, $c(v,u)$ – вес (в данном случае длина) ребра, $r_{\text{ср}}$ – средневзвешенный радиус (10)

4. Для полученных связанных компонент $G(V_i)$ могут быть применены различные метрики их включения в итоговый результат $G(V)$, например условие единственности штабеля

$$G(V) = \max|G(V_i)| \tag{24}$$

В текущей реализации было принято решение об исключении из рассмотрения групп из менее чем шести объектов:

$$G(V) = \cup\, G(V_i) : |G(V_i)| > 5 \tag{25}$$

Результат работы алгоритма на тестовых изображениях приведен на Рисунке 3.9.

а б в г д е

Рисунок 3.9 – Структура штабеля бревен, определенная графовым алгоритмом

3.2.3 Сегментация

Сегментация применяется для нахождения более точной границы каждого среза бревна и для выделения группы признаков, на основе которых будет производиться измерение объема бревен. На данном этапе производится уточнение контуров каждого среза исходя из окружностей, полученных для каждого объекта на предыдущих этапах и формирование атрибутов областей для дальнейшего измерения.

Анализ зарубежных и отечественных источников [66-68] показывает, что для сегментации изображений (разделение пикселей изображения на

переднеплановые и фоновые) широкое применение находят следующие группы методов

1. Методы, использующие анализ контуров изображений и основанные на моделях активных контуров и их разновидностях (змейка, двойная змейка, gradient vector flow и пр.) Основная идея этих методов основана на идее минимизации суммы некоторой внешней и внутренней энергий. Отличия методов – использование различных видов энергий [69-72].
2. Методы, использующие цветовую и текстурную сегментацию [73].
3. Методы, использующие теорию графов [73,74]. В таких методах изображения представляют в виде графа и производят его разрез, минимизирующий некоторую энергию, определяемую из целей сегментации.

Суть задачи нахождения минимального s-t разреза графа состоит в разбиении множества $G(V)$ вершин графа на два множества $G(S)$ и $G(T)$ $(G(S) \cup G(T) = G(V), G(S) \cap G(T) = \emptyset)$, где $s \in G(S), t \in G(T)$, таким образом, что сумма весов всех ребер ведущих из $G(S)$ в $G(T)$ минимальна:

$$\sum_{\substack{(i,j)\in G(E) \\ j\in G(S), j\in T}} c(i,j) \to min \qquad (26)$$

где $c(i,j)$ – вес ребра, соответствует разности яркостей пикселей i и j изображения.

Реализация минимального s-t разреза выполнена по алгоритму Бойкова-Колмогорова [75]

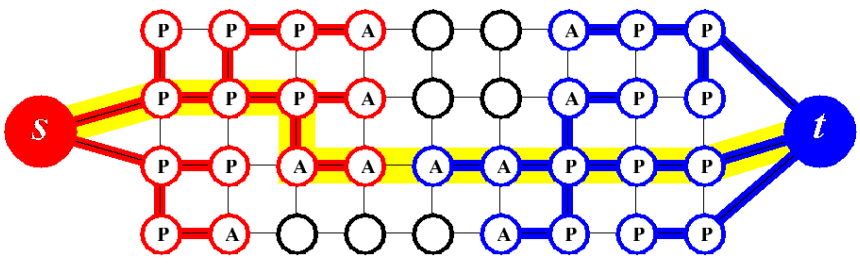


Рисунок 3.10 – s-t разрез

Узлы деревьев $G(S)$ и $G(T)$ могут быть активными и пассивными. Активная вершина соответствует границе дерева, пассивная – его внутренним элементам.

На начальном этапе в алгоритме указываются области источника (S) и истока (T) – терминальные вершины, а оставшиеся узлы графа остаются неразмеченными. Далее активные узлы присоединяют к своим графам дочерние из множества неразмеченных с наибольшим весом (и те в свою очередь становятся активными) итерационно до тех пор, пока не встречаются с размеченным узлом другого дерева. В этом случае можно говорить о том, что найден s-t путь (Рисунок 3.10).

Следующий этап – насыщение: часть ребер на предыдущем этапе стали насыщенными, вследствие чего часть узлов $G(S)$ и $G(T)$ стали «сиротами» – связывающие их с родителями ребра не действительны (они насыщенны). В результате деревья $G(S)$ и $G(T)$ расщепляются на лес: вершины-сироты формируют дополнительные вершины.

Третья стадия – адаптация – цель которой восстановление единичных деревьев $G(S)$ и $G(T)$. На этом этапе выполняется поиск нового родителя для созданных на предыдущем этапе вершин – «сирот». Новый родитель должен принадлежать к тому же множеству $G(S)$ или $G(T)$, что и «сирота» и связывающее ребро должно быть ненасыщенным. Если таковых не обнаружено, все дерево становится множеством неразмеченных вершин. Стадия завершается, когда не остается неразмеченных вершин и, таким образом, структура деревьев $G(S)$ и $G(T)$ восстановлена.

После стадии адаптации цикл повторяется и процедура завершается когда деревья $G(S)$ и $G(T)$ больше не могут расти, и они разделяются насыщенными ребрами. Это означает, что получен максимальный поток, что эквивалентно минимальному s-t разрезу [76].

В основе предложенного алгоритма сегментации срезов бревен лежит комбинация двух методов: алгоритм маркерного водораздела [77] и описанный выше алгоритм нахождения минимального s-t разреза в графе.

Алгоритм маркерного водораздела используется для задания областей уточнения торцов бревен. Результатом его выполнения будет множество сегментов изображения, включающих детектированные объекты радиальной

симметрии (Рисунок 3.11 а-е). При этом для каждого объекта известна его приблизительная граница – средневзвешенный радиус $r_{\text{ср}}$, который и требуется уточнить.

Используя имеющиеся входные данные для алгоритма минимального s-t разреза можно автоматически определить терминальные области S и T:

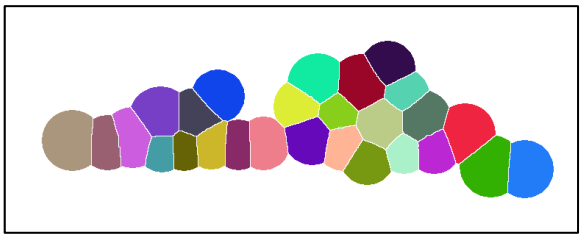
$$(S)_i \sim \left\{ s_i | s_i < \sqrt{\left(\frac{r_{\text{ср}_i}}{2}\right)^2 - d_i^2} \right\},\ (T)_i \sim \left\{ t_i | t_i > \sqrt{\left(\frac{3 \cdot r_{\text{ср}_i}}{2}\right)^2 - d_i^2} \right\}, \quad (27)$$

Результатом выполнения поиска минимального s-t разреза с учетом областей водораздела итоговый вид области, соответствующей срезу бревна, будет иметь вид:

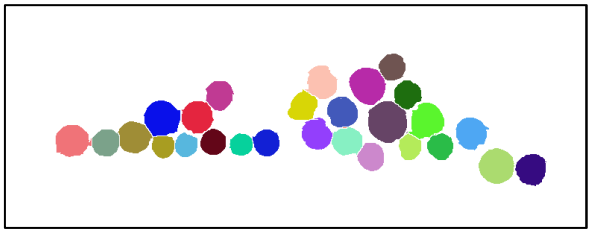
$$G(V)_i = G(S)_i \cap W_i \quad (28)$$

где $G(V)_i$ – область i-го среза бревна, $G(S)_i$ – область i-го среза бревна в результате s-t разреза, W_i – бассейн водораздела, включающий i-й маркер.

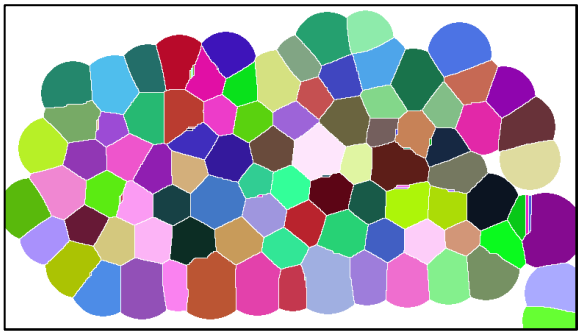
Результат работы алгоритма приведен на Рисунке 3.11,ж-м. На полученном изображении каждому пикселю присвоена метка объекта либо фона; пиксели, имеющие одинаковые цвета, отнесены к одному бревну.



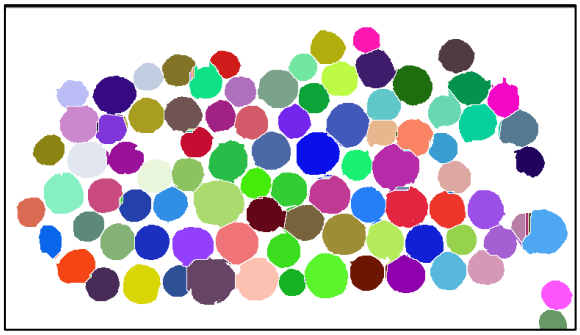
а



ж



б



з

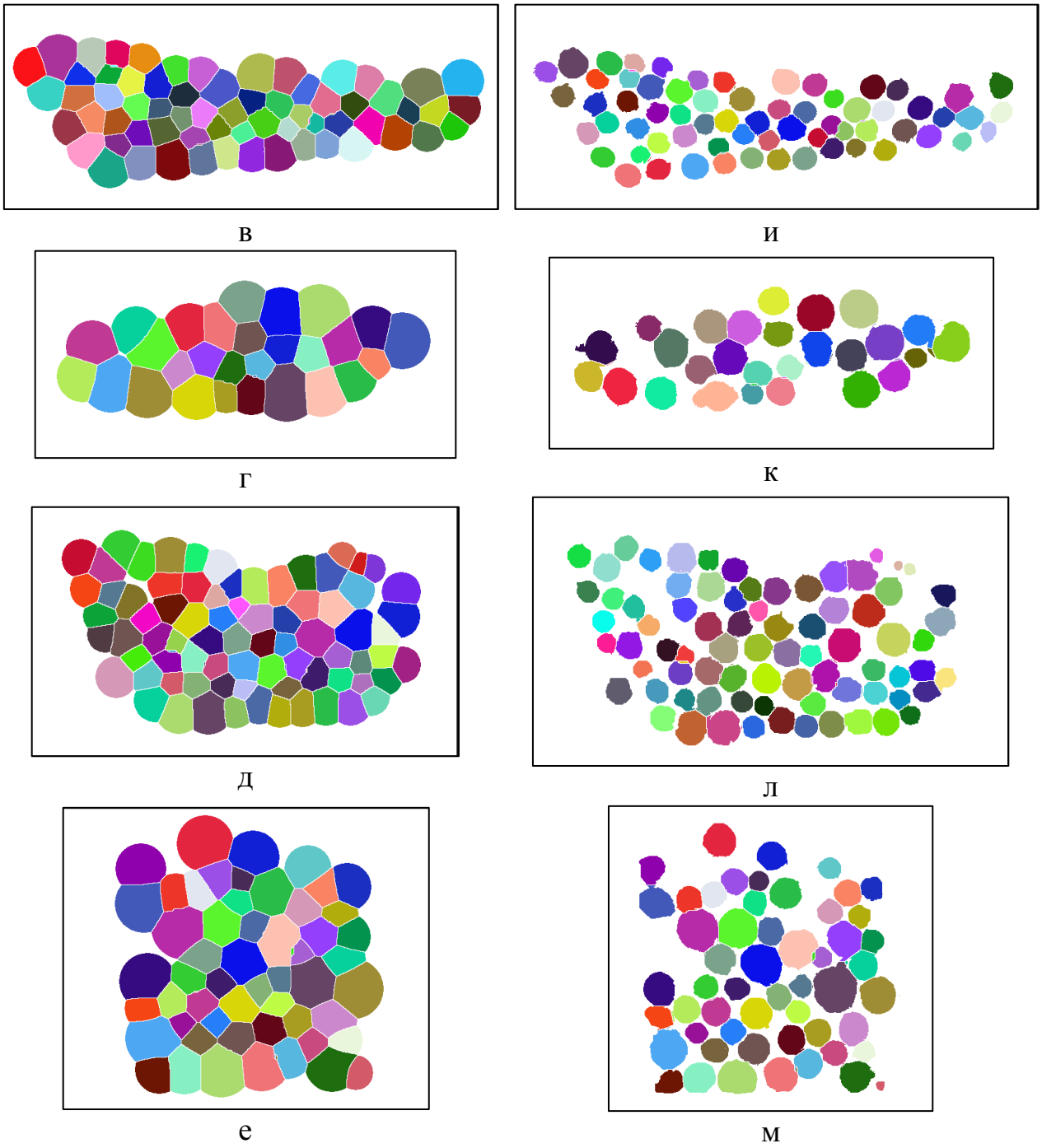


Рисунок 3.11 – Алгоритм сегментации бревен: а-е) метод водораздела; ж-м) результат работы

3.3 Сравнение результатов с ручным измерением

Основной особенностью круглых лесоматериалов является высокий уровень погрешностей при их учете. Измерения без погрешностей невозможны. Для круглых лесоматериалов предельные погрешности измерения объема и контроля качества устанавливают на уровне от ±3% до

±12%, что значительно превышает предельные погрешности учета для других видов материалов. Значительные погрешности учета круглых лесоматериалов обусловлены следующими основными причинами:

а) неправильной формой бревен и штабелей, значительными колебаниями плотности и влажности, наличием или обдиром коры;

б) погрешностями экспертной оценки учетчиками признаков (порода, пороки) и погрешностями визуального измерения показателей;

в) изменениями показателей и оцениваемых признаков с течением времени, их зависимость от условий произрастания древесины, особенностей производства и хранения круглых лесоматериалов.

Показанные на Рисунке 3.12 значительные отклонения объема и стоимости транспортных партий по результатам учета при приемке покупателем от результатов учета при отгрузке, также как и недостачи или излишки лесоматериалов на складе не являются следствием нарушения сохранности лесоматериалов при транспортировании или снижением их качества. Основная причина этих отклонений – погрешности учета [9].

Проблема измерений объемов круглого леса состоит в том, что из всей группы методов измерений нельзя выделить какой-либо один, дающий достоверный результат – все они имеют свою неустранимую погрешность, связанную со спецификой того или иного подхода.

На большинстве предприятий Свердловской области используется ручной поштучный метод измерения, поэтому сравнение разработанной методики оценки на основе фотограмметрии проведено с названным методом. Испытания проводилось на 940 изображениях, методика и результаты приведены в разделе 7 и Приложении Б. В данном разделе приведен сравнительный анализ результатов расчета объема штабеля круглого леса в случае автоматической обработки и ручного поштучного измерения на двух изображениях с целью определения степени достоверности получаемых результатов.

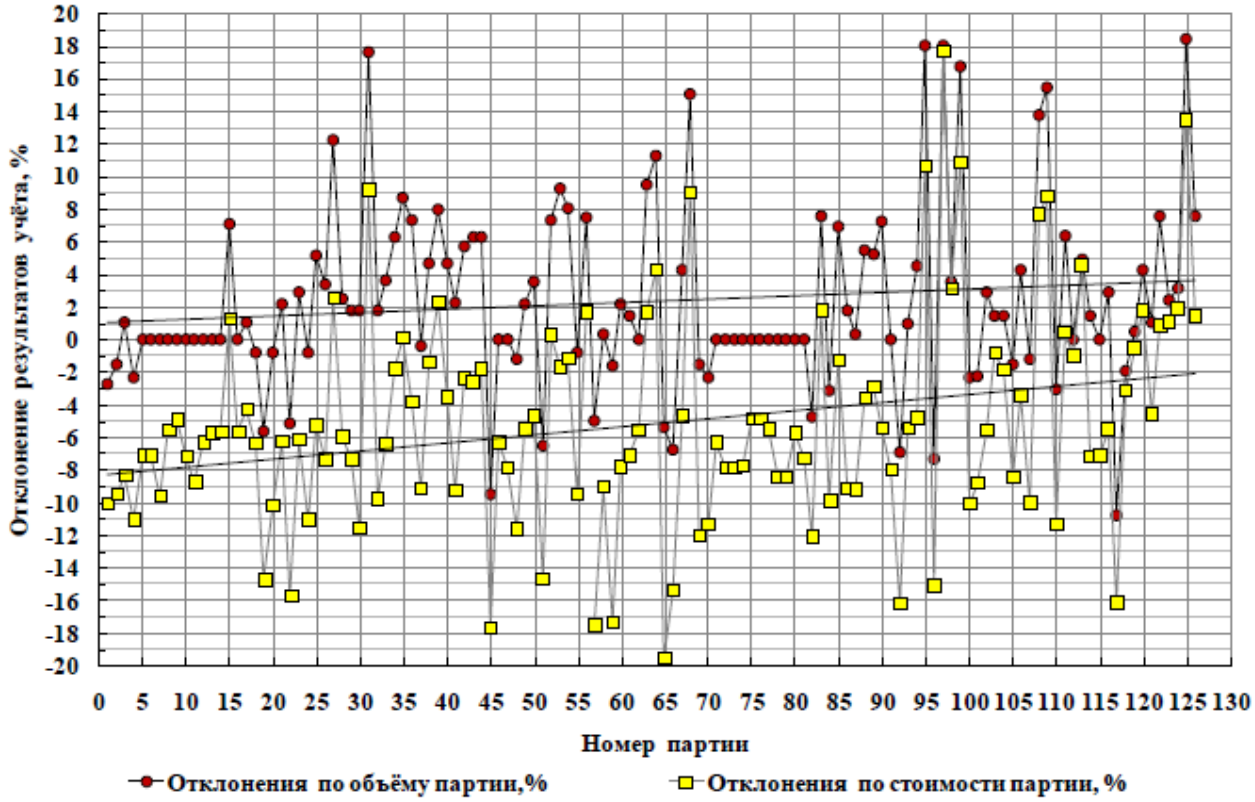


Рисунок 3.12 – Пример отклонений результатов учета при приемке от результата учета при отгрузке (в процентах от результата учета при приемке) для 126 автомобильных партий березового фанерного кряжа

На Рисунках 3.13-3.14 приведены примеры отчетов из программы для мобильной оценки объема штабеля круглого леса.

Далее рассмотрим каждое измерение по отдельности. На Рисунке 3.15 проведена нумерация бревен первого изображения, диаграмма на Рисунке 3.16 показывает результаты сравнения автоматического и поштучного измерения штабеля.

Категория	Имя	Значение
Изображение	Имя изображения	PC100041.JPG
	Время создания цифрового файла	2015:12:10 12:20:3073
Калибровка	Тип	По известному диаметру
	Диаметр калибровочного объекта, мм	240
	Коэффициент X, мм/пикс.	2,285714
	Коэффициент Y, мм/пикс.	2,376238
Кубатурник	Метод расчета	Таблица ГОСТ 2708-75
	Поправочный коэффициент на объем коры	1
	Таблица объемов	Таблица № 1
	Сбег бревен, см/м	1
	Длина штабеля, м	6,00
Результат	Количество, шт.	29
	Длина, м	6,00
	Объем, м³	12,65

Диаметр, см	Количество
24	8
26	7
28	6
30	3
32	4
34	1
Итого:	29

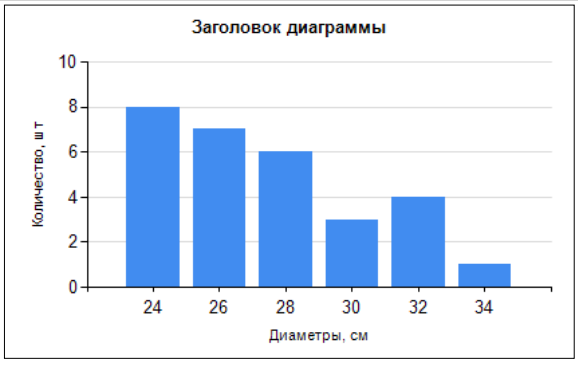


Рисунок 3.13 – Отчет от 12.24.2015 04:09:49

Категория	Имя	Значение
Изображение	Имя изображения	PC100042.JPG
	Время создания цифрового файла	2015:12:10 12:21:3174
Калибровка	Тип	По известному диаметру
	Диаметр калибровочного объекта, мм	220
	Коэффициент X, мм/пикс.	4,313725
	Коэффициент Y, мм/пикс.	4,074074
Кубатурник	Метод расчета	Таблица ГОСТ 2708-75
	Поправочный коэффициент на объем коры	1
	Таблица объемов	Таблица № 1
	Сбег бревен, см/м	1
	Длина штабеля, м	4,00
Результат	Количество, шт.	45
	Длина, м	4,00
	Объем, м³	8,62

Диаметр, см	Количество
20	5
22	21
24	15
26	4
Итого:	45

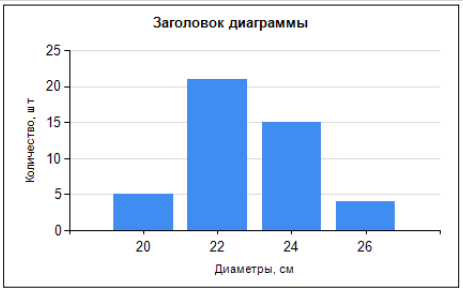


Рисунок 3.14 – Отчет от 12.29.2015 03:54:59

Рисунок 3.15. – Пронумерованные бревна

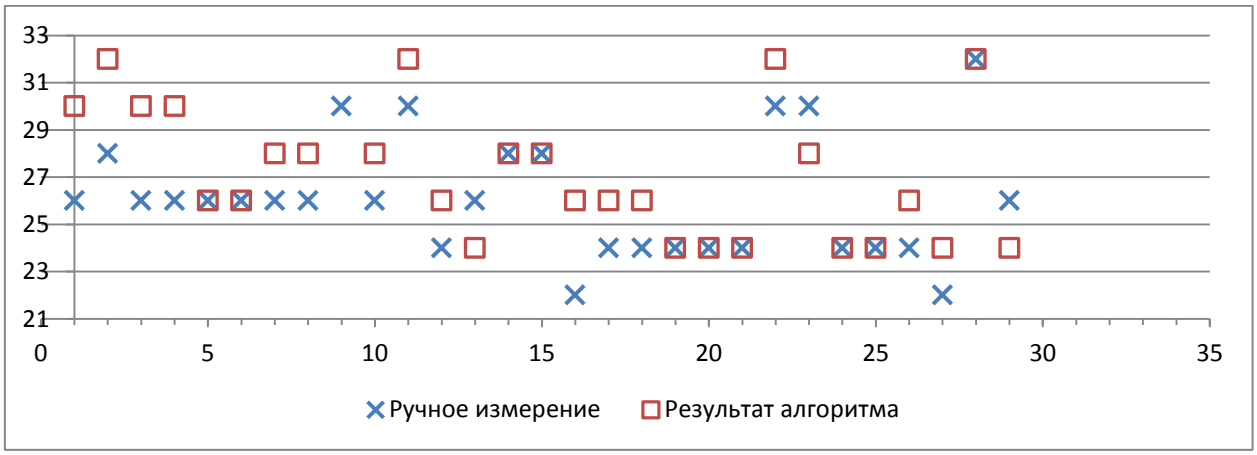


Рисунок 3.16 – Сравнение результатов измерения

Анализ достоверности результатов проведен путем наложения контуров бревен (Рисунки 3.17-3.21). Исходя из таблицы, диаметры бревен №№ 4 и 29 по ручному расчету эквивалентны – 26 см. Компьютерный анализ дает результат 30 см и 24 см соответственно. Сравнение контуров бревен показывает, что бревна № 4 имеет больший диаметр.

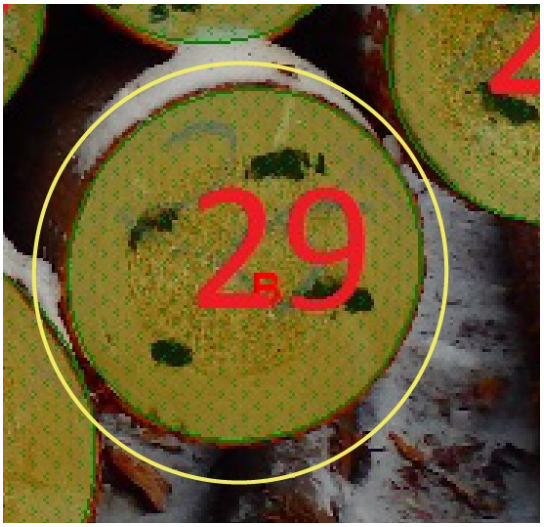


Рисунок 3.17 – Иллюстрация сравнения контуров торцов 4 и 29 бревен

Следующая пара бревен – №№ 13 и 27. В соответствии с ручным расчетом, их диаметры отличаются на 4 см. (26 и 22 см соответственно). По компьютерному расчету они равны – 24 см

Рисунок 3.18 – Иллюстрация сравнения контуров торцов 13 и 27 бревен

Пара бревен №№ 2 и 28. В соответствии с ручным расчетом, диаметры 2 и 28 бревен отличаются на 4 см. (28 и 32 см. соответственно). По компьютерному расчету они равны – 32 см

Рисунок 3.19 – Иллюстрация сравнения контуров торцов 2 и 28 бревен

Пара бревен №№ 5 и 16. По компьютерному расчету они равны – 26 см. По ручному – 26 и 22 см соответственно. На рисунке видно, что диаметр бревна № 5 действительно больше диаметра бревна № 16.

Рисунок 3.20 – Иллюстрация сравнения контуров торцов 5 и 16 бревен

Пара бревен №№ 10 и 23. По компьютерному расчету они равны – 28 см. По ручному – 26 и 30 см соответственно. Визуальная оценка показывает что фактическая разница диаметров +/- 2см.

Рисунок 3.21 – Иллюстрация сравнения контуров торцов 10 и 23 бревен

Вывод по первому изображению: визуально несоответствие результатов компьютерного расчета наблюдается только для бревен 5 и 16. В остальных случаях, для выбранных случаев наибольшего несоответствия (4 и более см разницы в расчетах) результаты компьютерного расчета представляются более достоверными.

На Рисунке 3.22 проведена нумерация бревен второго изображения, диаграмма на Рисунке 3.23 показывает результаты сравнения автоматического и поштучного измерения штабеля

Рисунок 3.22 – Пронумерованные бревна

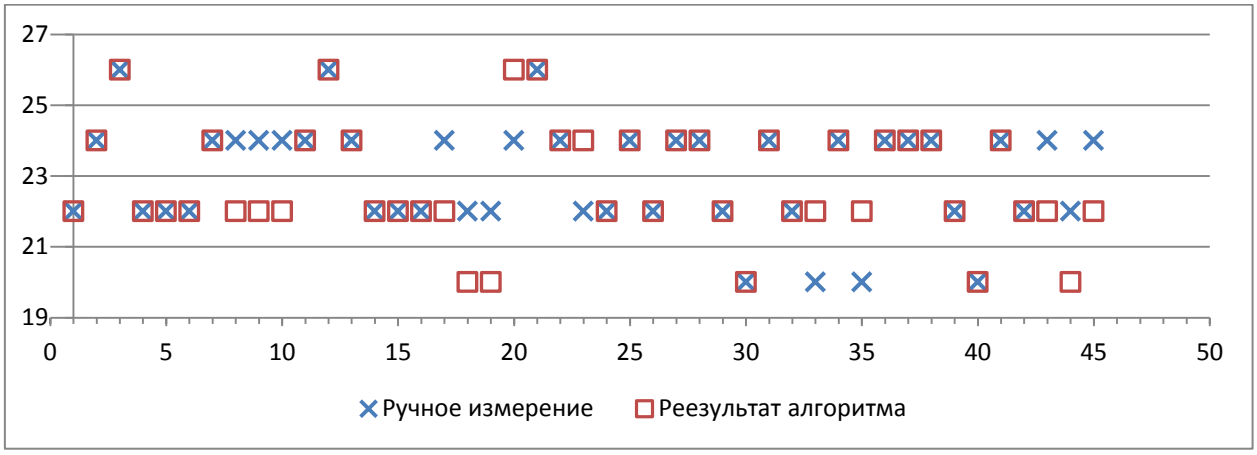


Рисунок 3.23 –Сравнение результатов измерения

Вывод по второму изображению: в основном результаты ручного и компьютерного расчета совпадают, разница в 2 см может являться следствием разницы расчета диаметра по оной (ручное измерение) или двум осям эллипса (компьютерный анализ) для каждого торца.

3.4 Выводы

Результаты тестирования показывают, что предложенный метод выделения объектов на основе модифицированного метода [59] достигает более высоких показателей качества по сравнению с методами, основанными на обучении линейных классификаторах (SVM+HOG, точность 77.9% [36]) и каскадах слабых классификаторов (AdaBoost+Haar, точность 95.1% при величине ложных срабатываний $4.9 \cdot 10^{-3}$). Следует также отметить, что предложенный способ превосходит методы основанные на преобразовании Хафа (CHT+LCM, точность 90.8%) (Таблица 3.1).

Таблица 3.1 – Сравнение методов выделения торцов бревен

Метод	TPR(σ_{TPR}), %	FPR
LBP [36]	95,8(3,7)	$2{,}4 \cdot 10^{-3}$
HOG [36]	77,9(10)	-
LBP+HOG [36]	96,0(3,1)	$2{,}4 \cdot 10^{-3}$
HAAR [36]	95,1(3,8)	$4{,}9 \cdot 10^{-3}$

Метод	TPR(σ_{TPR}), %	FPR
HAAR+HOG [36]	96,0(3,5)	$4{,}9 \cdot 10^{-3}$
CHT+LCM [34]	90,8(9,4)	-
CHT [34]	84,8(11,6)	-
LCM [34]	89,7(9,8)	-
Предложенный метод	96,2(4,1)	-

Детальный анализ результатов работы выполнен на примере двух изображений и проведено сравнение предлагаемого алгоритма с ручным измерением. Визуальная оценка для случаев расхождения более чем на 4 см показала, что погрешность присутствует в обоих случаях. Компьютерный анализ наиболее чувствителен к торцам сильно эллипсоидной формы (Рисунок 3.15, бревно 5). Зависимость расхождения результатов от оптических искажений не выявлена (другими словами, нет ситуации, при которой в центре изображения значения диаметров совпадали, а ближе к краям – расходились). Для второго случая расхождения находятся в пределах 2 см; такая погрешность может возникнуть из-за различных подходов к измерению диаметра: одно измерение на торце в случае поштучного измерения и расчет среднего арифметического двух осей эллипса в случае компьютерного анализа.

ГЛАВА 4. ТРЕБОВАНИЯ К ИСХОДНЫМ ДАННЫМ

4.1 Выбор конфигурации съемки

Качество производимых измерений на изображении существенно зависит от выбранного масштаба съемки, расположения съемочных датчиков, а так же от ограничений, налагаемых на условия съемки. Исходя из геометрической структуры штабеля бревен, для создания бесконтактной измерительной системы был предложен способ, обеспечивающий, на наш взгляд, наиболее оптимальный по эффективности и трудозатратам метод. Способ заключается в фотографировании двух сторон штабеля бревен с дальнейшей оцифровкой и обработкой фотографий специализированным программным обеспечением для выделения всех торцов бревен каждого изображения, восстановления трехмерной пространственной структуры штабеля и определения его геометрических характеристик, таких как объем, диаметр вершины и комля бревен, сбег каждого бревна, суммарный объем и средний сбег штабеля. Единственным параметром, который необходимо вводить в программу, является длина измеряемых бревен. При отсутствии возможности фотографирования штабеля с двух сторон допускается производить съемку только одного торца, при этом построение модели и расчет некоторых параметров штабеля будет производиться исходя из принятых математических моделей при заданных допущениях и ограничениях. Например, при отсутствии априорных данных о сбеге данный параметр необходимо вводить пользователю.

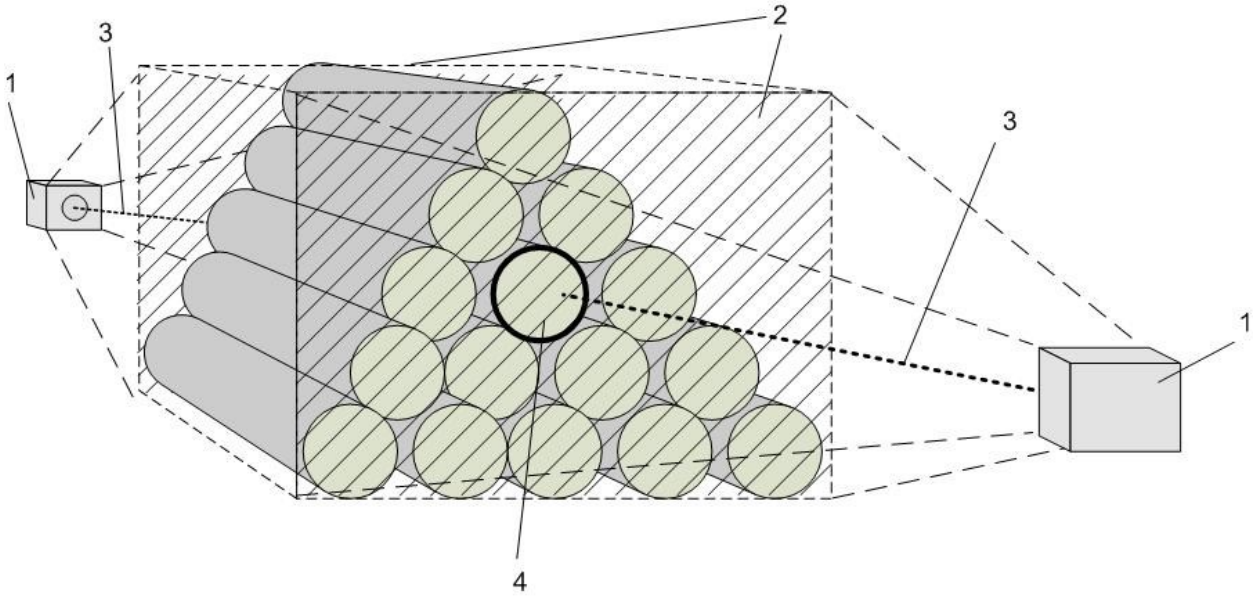


Рисунок 4.1 – Схема съемки штабеля бревен. 1 – цифровое устройство съема изображений; 2 – оптическая плоскость; 3 – главная оптическая ось; 4 – эталонноый объект

4.2 Калибровка сенсоров

Под задачей калибровки понимается определение адекватной модели съемки и оценка ее параметров для определения пространственных координат объектов по их изображениям. В фото и видеокамерах, использующихся в системах машинного зрения, снимок формируется как перспективное отображение трехмерных точек на плоскость двумерного изображения, когда каждая точка снимка определяется как точка пересечения прямой, походящей через центр проекции и точку объекта с плоскостью снимка.

В теории фотограмметрии для описания геометрической модели камеры вводят две системы координат: пиксельную (X_i,Y_i), связанную с оцифрованным изображением (единицей измерения в этой системе координат является пиксель) и пространственную систему координат цифрового снимка (X,Y,Z).

Переход между системами, определяющий связь пиксельной и пространственной систем координат, может быть задан следующими выражениями:

$$\begin{aligned} x_i &= (f/z) \cdot x \\ y_i &= (f/z) \cdot y \end{aligned} \tag{29}$$

где x,y,z – пространственный размер и расстояние до объекта, x_i,y_i – пиксельный размер образа объекта, f – фокусное расстояние объектива датчика.

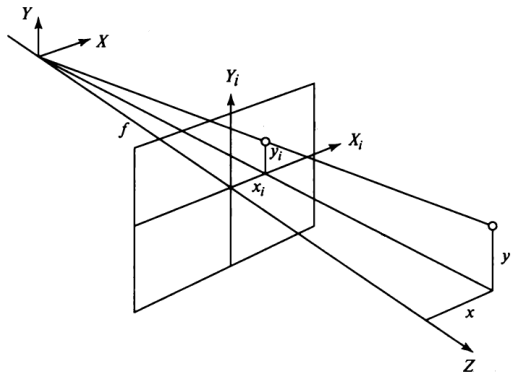


Рисунок 4.2 – Модель перспективной проекции для получения изображения

Такая математическая модель может использоваться для вычисления размеров торцов бревен при условии, что с помощью камеры наблюдается плоский объект, расположенный на фиксированном расстоянии от датчика [66]. В таком случае изображение объекта будет выглядеть как копия этой плоскости в уменьшенном масштабе. Следовательно, при таком взаимном расположении объекта измерения и камеры можно будет работать непосредственно в системе координат изображения, считая изображение масштабированной копией пространственного объекта. В самом деле, если при съемке торцов бревен расположить камеру параллельно штабелю и ввести допущение, что все торцы бревен лежат в одной плоскости на фиксированном расстоянии от датчика, то размеры срезов бревен могут быть вычислены по следующим формулам:

$$\begin{aligned} x &= m_x \cdot x_i \\ y &= m_y \cdot y_i \end{aligned} \tag{30}$$

где m_x, m_y – масштабные коэффициенты, определяющие переход от номеров строк/столбцов изображения к миллиметрам.

4.3 Выводы

Под задачей калибровки понимается определение адекватной модели съемки и оценка ее параметров для определения пространственных координат объектов по их изображениям. Поскольку качество производимых измерений существенно зависит от расположения съемочных датчиков, при проведении съемок штабеля необходимо располагать фотокамеру параллельно плоскостям, образуемым торцами бревен. При таком взаимном расположении объекта измерения и камеры можно определять параметры бревен непосредственно в системе координат снимка.

Таким образом, для определения размеров торцов бревен необходимо знать масштабные коэффициенты m_x, m_y, определяющие ширину и высоту пикселя изображения. Для этого можно произвести распознавание шаблонного объекта на изображении, а при обработке результатов ввести в программу заранее измеренные его размеры для получения коэффициентов пересчета. Единственное ограничение, предъявляемое к калибровочному объекту, он должен располагаться в плоскости срезов бревен и быть доступным для измерения программными средствами.

ГЛАВА 5. ИЗМЕРЕНИЕ ОБЪЕМА ШТАБЕЛЯ КРУГЛОГО ЛЕСА

5.1 Построение модели пакета бревен

В результате проведенных операций распознавания и калибровки возможно выполнить реконструкцию трехмерных координат целевых объектов полученных по изображениям двух торцов штабеля. При этом, для того чтобы вычислить точное положение каждого бревна в штабеле необходимо определить его положение на каждом из снимков. Другими словами, необходимо установить однозначное соответствие интересующего нас среза каждого бревна на изображении, полученном с одной точки съема, его же изображению на снимке, полученном с другой стороны штабеля. Решение такой задачи позволит определить адекватную модель штабеля бревен для точного определения геометрических характеристик круглых лесоматериалов. Другими словами, для каждого бревна в штабеле будет получена информация о его ориентации и для каждого торца на изображении будет определено, является ли он комлем или вершиной бревна

Математическая модель штабеля бревен для данной задачи может быть определена как совокупность усеченных конусов, где большее основание является комлем, меньшее – вершиной бревна. Пространственную структуру данной модели и нужно определить на данном этапе. Если сформулировать задачу совмещения двух изображений срезов бревен как задачу оптимизации и ввести функционал, минимизация которого обеспечит наилучший результат сопоставления, эта задача может быть сведена к одной из фундаментальных задач в области математической оптимизации – задаче о назначениях – и решена с применением аппарата комбинаторной оптимизации [78]. Задача формулируется следующим образом:

Даны два множества, U и V, одного размера и задана функция стоимости C. Необходимо сопоставить каждому элементу одного множества ровно один элемент другого множества $f: U \to V$, таким образом, что

$$\sum C(u, f(u)) \rightarrow min \quad (31)$$

В условиях данной задачи потребуем, чтобы сумма всех евклидовых расстояний между парой срезов двух изображений была минимальна. Тогда определяя структуру данных в терминах двудольного графа, результатом алгоритма будет является список ребер, направленных от U к V и образующих паросочетание с наименьшей стоимостью, по которому строится модель.

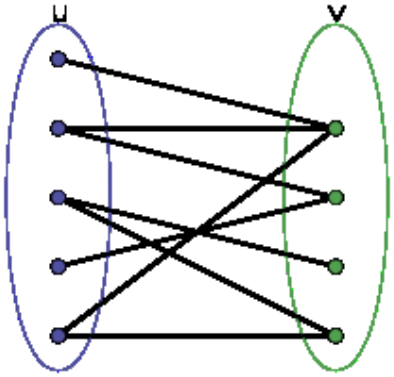


Рисунок 5.1 – Иллюстрация к задаче о назначениях

В случае, когда имеется только одна точка съема, необходимо решить задачу определения трехмерной структуры штабеля по одному изображению. В условиях отсутствия некоторых ключевых параметров для расчета объема, а именно информации об ориентации и сбеге бревен, исходными данными для построения модели являются области срезов бревен одного изображения, а недостающие параметры определяются согласно принятому предположению о том, что сумма площадей торцов одной стороны штабеля примерно соответствует сумме площадей торцов другой стороны. Для разделения всех торцов на множества комлей и вершин выполняется следующая процедура:

1. Все выделенные торцы сортируются по размеру.
2. Ориентация бревен равновероятна, значит, количество вершинных торцов соответствует количеству комлевых (если в параметрах измерения не указано иное). Для соблюдения равенства сумм площадей торцов четным

элементам отсортированного множества присваивается метка вершины, нечетным – комля.

3. В случае если видимый торец – комель, для определения диаметра вершины бревна используется коэффициент сбежистости:

$$d = D - s \cdot L \tag{32}$$

где d – диаметр вершины бревна, D – диаметр комля, s – коэффициент сбежистости (нормальный коэффициент сбежистости равен 1 см/м), L – длина штабеля.

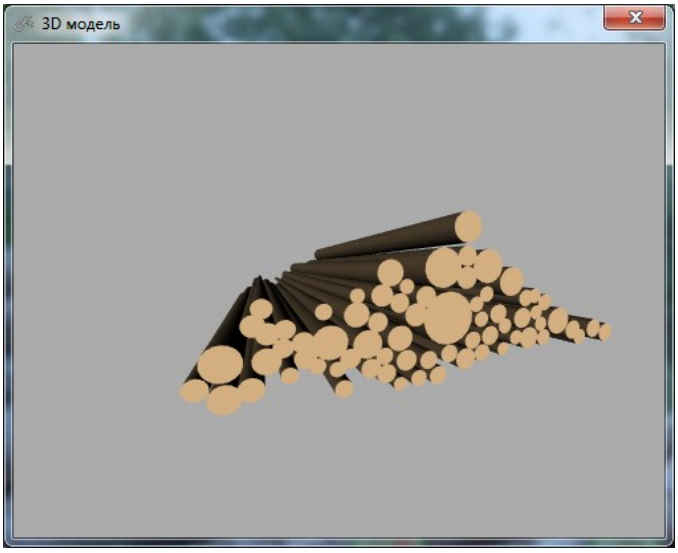


Рисунок 5.2 – Пример модели пучка бревен, построенной по одному изображению

5.2 Измерение объема штабеля круглого леса

После того, как на изображении обнаружено множество размеченных областей требуется определить их свойства, которые будут использоваться в качестве входных данных для измерения объема. Рассчитываются следующие геометрические характеристики: площадь, геометрический центр, ориентация, длина и направление полуосей эквивалентного эллипса. Для вычисления указанных характеристик используется математический аппарат теории моментов инерции материального тела [67].

Площадь определяется как количество пикселей области

$$A = \sum\sum I(x, y) \tag{33}$$

Центр масс рассчитывается из следующих выражений

$$\bar{x} = \frac{\sum\sum x \cdot I(x,y)}{A}$$
$$\bar{y} = \frac{\sum\sum y \cdot I(x,y)}{A} \quad (34)$$

направление полуосей эллипса определяется собственными векторами матрицы ковариации, а собственные числа определяют длину полуосей.

$$cov = \begin{pmatrix} m_{20} & m_{11} \\ m_{11} & m_{02} \end{pmatrix} \quad (35)$$

$$\theta = \frac{1}{2} atan\left(\frac{2m_{11}}{m_{20} - m_{02}}\right) \quad (36)$$

где дискретный центральный момент m_{ij} определяется следующим образом:

$$m_{ij} = \sum (x - \bar{x})^i (y - \bar{y})^j I(x,y) \quad (37)$$

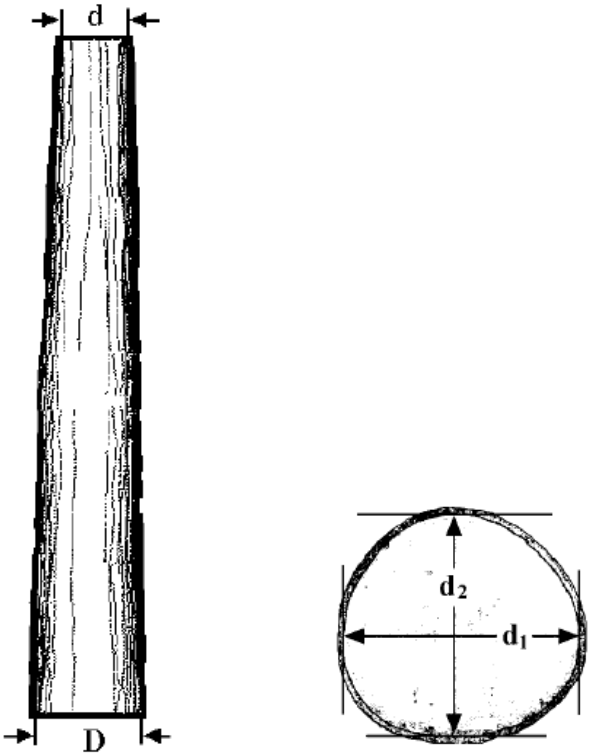
Измерение объема штабеля круглого леса производится исходя из требований ГОСТ 32594-2013 и на основании введённой ранее математической модели штабеля, согласно которой объект измерения определяется как совокупность усеченных конусов, а их основания, доступные для измерения, описываются через эквивалентные эллипсы.

Диаметр в верхнем и нижнем торце каждого бревна рассчитывается как среднеарифметическое значение двух взаимно перпендикулярных диаметров, заданных малой и большой осями эллипса.

$$d = \frac{d_1 + d_2}{2} \quad (38)$$

Объем всего пакета рассчитывается как сумма объемов отдельных бревен, образующих пакет по формуле:

$$V = \sum_{i=0}^{n} V_i \quad (39)$$



а) б)

Рисунок 5.3 – Пояснение к задаче измерения: а) расположение верхнего и нижнего диаметров бревна; б) измерение верхнего и нижнего диаметра на торце

Измерение объема каждого бревна производится с использованием следующих методов:

Метод срединного сечения предусматривает вычисление объема цилиндра, основанием которого служит площадь поперечного сечения, взятого на середине бревна. Вычисление объема бревна V, м^3, проводят по формуле

$$V = 3{,}1416 \cdot d_c^2 \cdot L/4 \cdot 1000 \quad (40)$$

где d_c – срединный диаметр бревна, см;

L – длина бревна, м.

Метод усеченного конуса предусматривает измерение верхнего диаметра d, нижнего диаметра D и длины бревна L. Вычисление объема бревна проводят по формуле усеченного конуса.

$$V = \frac{3{,}1416 \cdot L(d^2 + D^2 + d \cdot D)}{12 \cdot 10000} \quad (41)$$

где d – диаметр бревна в верхнем торце, см;

D – диаметр бревна в нижнем торце, см;

L – длина бревна, м.

Метод верхнего диаметра и среднего сбега предусматривает определение объема бревна умножением площади поперечного сечения на середине длины бревна на его длину. С учетом технологичности процесса измерения верхнего диаметра бревна по сравнению с измерением срединного диаметра бревна проводят пересчет верхнего диаметра d в срединный диаметр d_c, по формуле:

$$d_c = d + s \cdot \frac{L}{2} \quad (42)$$

где d– верхний диаметр бревна, см;

s – сбег бревна, см/м;

L – длина бревна, м.

Вычисление объема бревна V, $м^3$, проводят по формуле

$$V = \frac{3{,}1416 \cdot L \cdot (d + s \cdot \frac{L}{2})^2}{4 \cdot 10000} \quad (43)$$

Метод концевых сечений предусматривает определение объема бревна по измерениям верхнего диаметра d, нижнего диаметра D и длины бревна L. Определение объема бревна основано на усреднении объемов двух цилиндров с диаметрами оснований, равными диаметрам торцов. Объем бревна V, $м^3$, вычисляется по формуле:

$$V = \frac{3{,}1416 \cdot L \cdot (d^2 + D^2)}{8 \cdot 10000} \quad (35)$$

Метод определения объема круглых лесоматериалов по таблицам ГОСТ 2708. Объем круглых лесоматериалов определяют измерением диаметра верхнего торца бревен без коры и их длины и нахождением по этим параметрам объема древесины без коры по таблицам ГОСТ 2708.

При этом для случаев анализа по одному изображению могут использоваться только метод верхнего диаметра и среднего сбега и ГОСТ 2708.

Кора из объема круглых лесоматериалов исключается умножением объема бревна, измеренного с корой V_k, на поправочный коэффициент на объем коры P_k:

$$V = V_k \cdot P_k \quad (44)$$

Поправочный коэффициент на объем коры определяет пользователь на основе визуальной оценки изображений штабеля бревен.

5.3 Выводы

Предложенный метод реализован в алгоритме измерения объема круглого леса, который был также протестирован на 940 изображениях штабелей леса:

– в штабелированном виде и на сортиментовозе;

– ориентированными к фотообъективу комлями, торцами и враскомлевку;

– в различных погодных условиях (снег, солнечная погода).

Рассчитанная по итогам испытаний средняя относительная ошибка определения объема в результате работы алгоритма автоматического выделения и измерения торцов бревен не превосходит 7.1%, что является лучшим показателем по сравнению с [34] (Таблица 5.1).

Таблица 5.1 – Сравнение методов расчета объема штабеля круглого леса

Метод	средняя погрешность %
CHT+LCM [34]	14,22
CHT [34]	29,06
LCM [34]	21,42
Предложенный метод	7,1

ГЛАВА 6. ПРОГРАММА ДЛЯ МОБИЛЬНОГО РАСЧЕТА ОБЪЕМА ШТАБЕЛЯ КРУГЛОГО ЛЕСА

Программа для мобильного расчета объема штабеля круглого леса предназначена для бесконтактного измерения объема и геометрических характеристик уложенного в пакеты круглого леса посредством фотограмметрического анализа изображений торцов бревен и предоставления результатов измерения оператору программы в виде отчета.

Программа реализует алгоритм, предложенный в предыдущем разделе.

Программа предполагает эксплуатацию в условиях предприятий лесопромышленного комплекса в местах заготовки круглого леса, в пунктах приема/сдачи грузов и на складах. Конечными пользователями программы являются сотрудники профильных подразделений лесопромышленного предприятия.

6.1 Назначение программы

Программа обеспечивает выполнение следующих основных функций:

– *загрузку одного или нескольких изображений.* Количество изображений определяется доступным для фотографирования числом точек съема штабеля. Допустимое количество анализируемых изображений задается в настройках программы;

– *калибровку изображений.* Калибровка заключается в определении внешних и внутренних параметров камеры. Калибровка выполняется в ручном режиме оператором программы в порядке, описанном в документе «Руководство оператора»;

– *автоматический поиск торцов бревен.* Выполняется без участия оператора программы. Качество работы автоматического выделения зависит от степени соответствия входных данных требованиям, описанным в настоящем руководстве;

– *ручной поиск и редактирование торцов бревен.* Ручная корректировка результатов может применяться в случаях, когда автоматический алгоритм допустил ошибку, например, если имеет место частичное перекрытие бревен.

– *расчет объема пакета.* Расчет объема осуществляется в соответствии с действующими нормативными документами;

– *получение и сохранение результата.* Результат работы программы отражается в детальном отчете с возможностью его экспорта в различные форматы (DOC, XLS, PDF) или вывода на печать.

6.2 Описание логической структуры

Программное обеспечение состоит из следующих частей:

– Основная исполняемая программа Forest.exe. Включает пользовательский интерфейс и методы цифровой обработки изображений для поиска срезов бревен на изображении;

– Вспомогательная программа ForestHelp.exe. Вызывается основной программой и содержит справочную информацию по работе с программой;

– Библиотека ForestCubaturnic.dll. Реализует методы расчета объема в соответствии с действующими нормативными документами (ГОСТ 32594-2013 [79] и ГОСТ 2708-75 [80]);

– Библиотека ForestReporting.dll. Включает методы и классы для просмотра отчетов, созданных с применением технологии отчетности Microsoft.

Все программные компоненты реализованы на языке высокого уровня C++/CLI.

Работа с программой предполагает совершение определенной последовательности действий для получения требуемого результата. Схема последовательности действий по работе с основной программой включает следующие этапы.

4. загрузка изображений;

5. калибровка изображений;
6. автоматический поиск бревен;
7. ручное редактирование;
8. измерение торцов бревен;
9. получение и анализ результата.

6.2.1 Структура программы

Логическая структура программы состоит из пяти визуальных и тринадцати расчетных модулей.

Интерфейсная часть программы содержит:

- модуль основного окна программы;
- модуль окна настроек программы;
- модуль окна навигатора;
- модуль просмотра и сохранения отчетов;

Расчетная часть программы включает:

- модуль алгоритма;
- модуль настроек программы;
- модуль констант;
- модуль методов расчета объема;
- модуль загрузчика изображений;
- модуль математических функций;
- модуль модели;
- модуль настроек модели;
- модуль формирования отчетов;
- модуль настроек визуализации результатов;
- модуль контроллера;
- модуль графических примитивов.

6.2.1.1 Модуль основного окна программы

В модуле описан класс MainForm основного окна программы Forest.exe. Класс является визуальным компонентом. Отвечает за отображение информации (визуализацию) и взаимодействие с пользователем. В качестве представления выступает форма с графическими элементами.

6.2.1.2 Модуль настроек программы

В модуле описан один класс SettingsForm вспомогательного окна программы Forest.exe. Класс является визуальным компонентом. Отвечает за отображение и редактирование пользовательских настроек основной программы.

6.2.1.3 Модуль окна навигатора

В модуле описан класс вспомогательного окна программы Forest.exe. Класс является визуальным компонентом. Отвечает за отображение и позиционирование изображения в основном окне изображения. Дублирует область вывода изображения основного окна программы в уменьшенном масштабе.

6.2.1.4 Модуль просмотра и сохранения отчетов

В модуле описан класс вспомогательного окна программы Forest.exe. Класс является визуальным компонентом. Отвечает за отображение отчета программы. Модуль реализует механизмы: просмотра, редактирования, сохранения и печати отчета. В модуле реализована возможность сохранения отчетов в форматах pdf, doc, xsl.

6.2.1.5 Модуль алгоритма

В модуле реализованы более двадцати классов для реализации алгоритмов выделения торцов бревен на изображении. Основу составляют следующие классы:

1. ManualAlgoritm – класс ручного выделения срезов бревен

2. AutoAlgoritm – класс автоматического выделения срезов бревен
3. ManualCalibrateAlgoritm – класс выделения калибровочного объекта
4. EmptyAlgoritm – класс пустого алгоритма (добавлен для совместимости)
5. IBackgroundAction – интерфейсный класс. Реализуется в классах ManualAlgoritm, AutoAlgoritm, ManualCalibrateAlgoritm, EmptyAlgoritm.

6.2.1.6 Модуль настроек программы

Модуль настроек программы содержит один класс. Предоставляет интерфейс для изменения пользовательских данных в окне настроек программы. Содержит следующие поля:

1. настройки модели;
2. настройки отображения результатов.

6.2.1.7 Модуль констант

В модуле описаны константные выражения, используемые алгоритмом основной программы.

6.2.1.8 Модуль методов расчета объема

Модуль содержит пять классов для реализации методов расчета объема:

1. ICubaturnic – абстрактный класс таблицы-кубатурника
2. ColomnBase – базовый класс столбца таблицы-кубатурника
3. TableBase – базовый класс табличных методов расчета объема
4. Column2708_75_60 – класс столбца из госта 2708-75
5. TableGOST_2708_75 – класс, реализующий таблицу-кубатурник (гост 2708-75)

Целесообразность выбора такой структуры обусловлена следующим:

– Максимальная абстракция, что позволяет добавлять новые методы расчета объема к программе не меняя основного функционала;

– Все элементы наследуют один класс, это делает простым добавление новых свойств в настройки программы.

6.2.1.9 Модуль загрузчика изображений

В модуле загрузчика изображений реализован один класс ImageLoader. Содержит поля и методы для чтения файла изображения с жесткого диска, изменения размеров и привидения к одному типу. В классе реализована возможность считывания и преобразования 8, 24 и 32 битных изображений формата bmp, jpg, jpeg. Входными данными для модуля является файл изображения, выходными – три массива изображения, соответствующие красной, зеленой и синей компонентам изображения.

6.2.1.10 Модуль математических функций

В модуле реализованы структуры и математические методы для реализации алгоритма детектирования, выделения и расчета штабеля круглого леса на изображении.

6.2.1.11 Модуль модели

Модуль содержит классы моделей программы. Предоставляет данные и методы для работы всей логики программы, реагирует на запросы от класса контроллера, изменяя своё состояние, возвращает данные (результаты работы алгоритма, пользовательские уведомления, ошибки) для отображения, при этом не содержит информации, как эти данные можно визуализировать.

6.2.1.12 Модуль настроек модели

Модуль настроек модели содержит один класс. Предоставляет интерфейс для изменения пользовательских данных в окне настроек программы. Содержит следующие поля:

- настройки методов расчета объема
- наименьший диаметр поиска
- наибольший диаметр поиска
- диаметр калибровочного объекта

6.2.1.13 Модуль формирования отчетов

Модуль формирования отчетов реализует следующие классы:

1. ReportField – класс реализует одно поле таблицы отчета;
2. ReportString – класс реализует строку отчета. Объекты класса представляют заголовок, изображение и служебные поля отчета;
3. ReportViewData – класс представляет статистические данные отчета.

6.2.1.14 Модуль настроек и визуализации

Модуль настроек модели содержит нескоько классов. Предоставляет интерфейс для изменения отображаемых данных (результата работы алгоритма) в окне настроек программы. Содержит кисти для рисования навигатора, срезов бревен и выпуклой оболочки.

6.2.1.15 Модуль контроллера

Модуль реализует один класс. Обеспечивает связь между пользователем и системой: контролирует ввод данных пользователем, использует модель и модули отображения для реализации необходимой реакции программы.

6.2.1.16 Модуль графических примитивов

Реализует несколько классов графических примитивов для отрисовки результатов работы программы в главном окне. Используется для вывода на экран:

- каждого найденного среза;
- области интереса;
- прямоугольника навигатора.

6.3 Алгоритм работы программы

Основной алгоритм программы представлен блок-схемой на Рисунке 6.1.

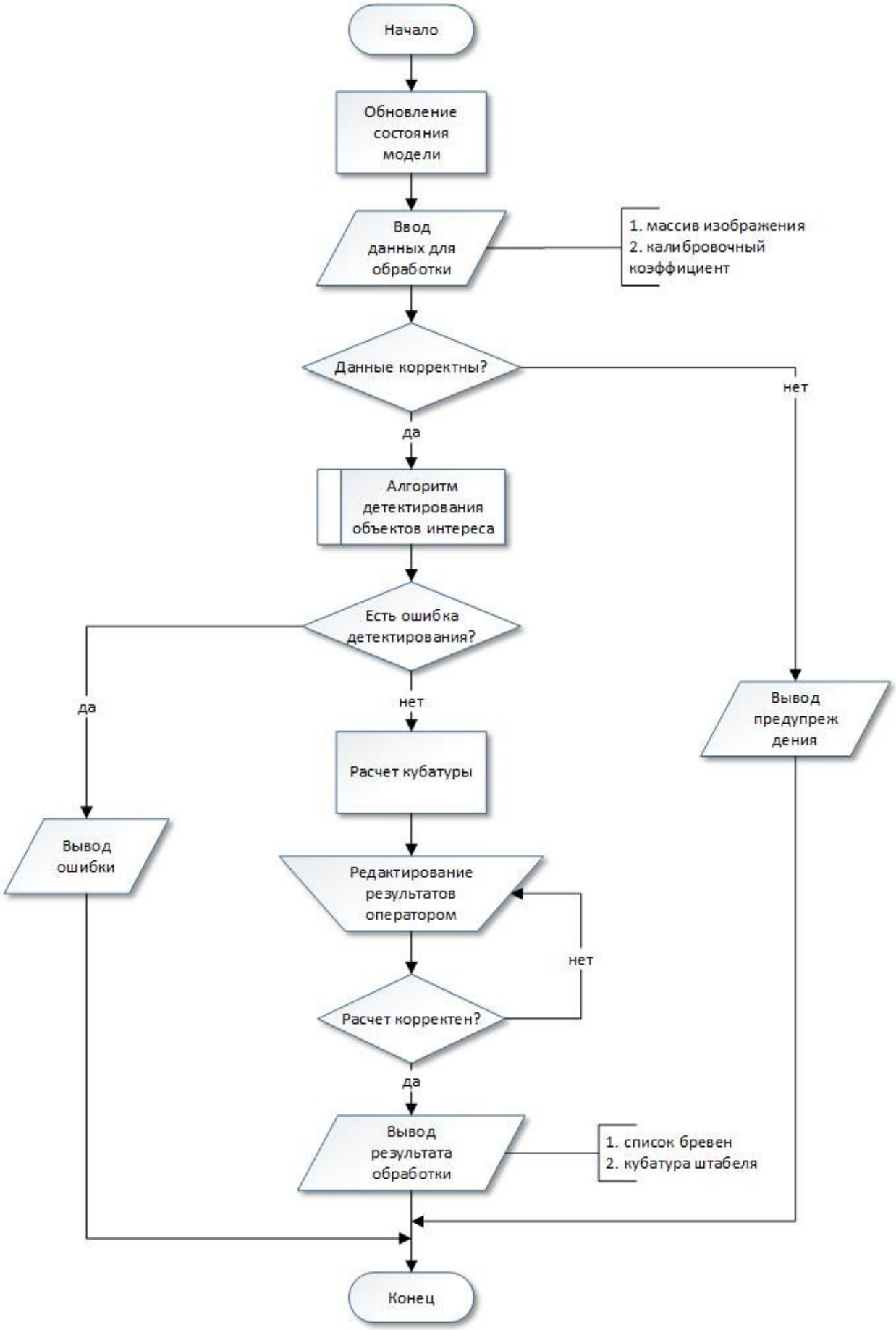


Рисунок 6.1 – Блок-схема алгоритма программы

Описание основного алгоритма программы:

1. Обновление состояния модели – все поля модели приводятся в первоначальное состояние для старта алгоритма;
2. Очищение списка отображения найденных срезов;
3. Запуск алгоритма выделения срезов бревен;
4. Формирование списка найденных бревен;
5. Расчет объема;
6. Отображение результата работы алгоритма.

6.4 Интерфейс программы

6.4.1 Главное окно программы

Главное окно программы предназначено для управления работой всего программного обеспечения (см. Рисунок 6.2).

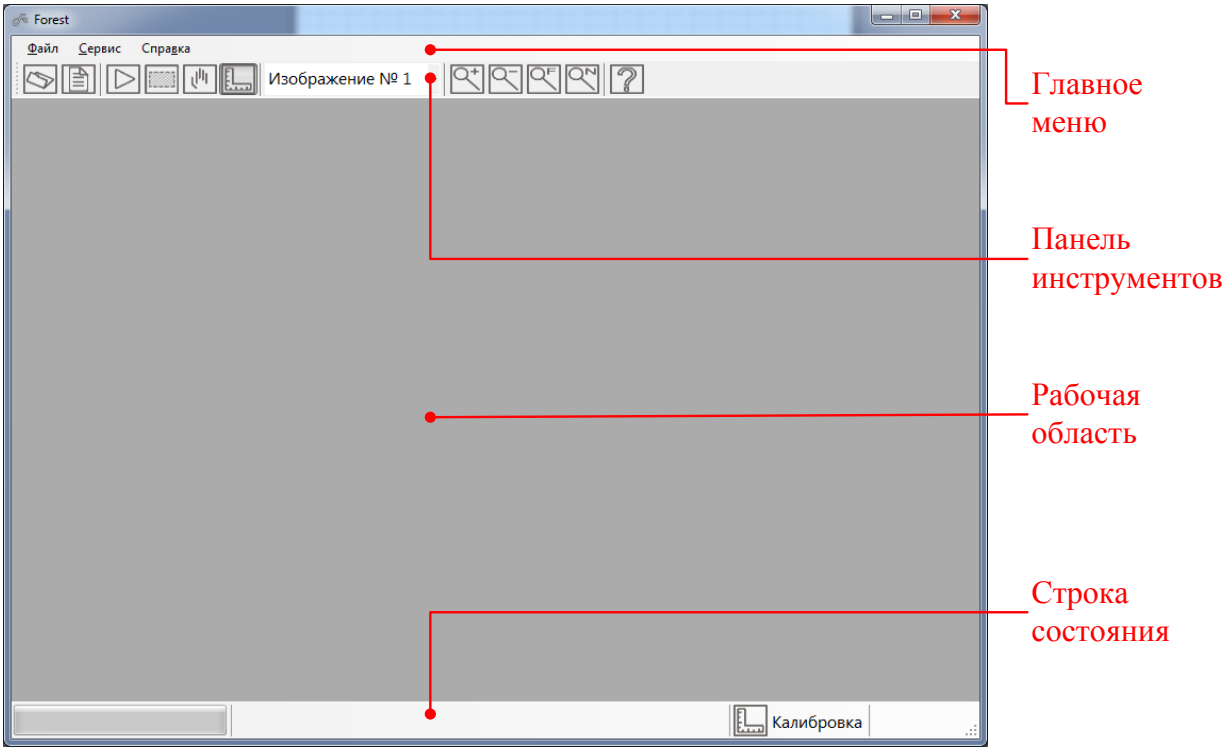


Рисунок 6.2 – Интерфейс главного окна программы

Функции основных элементов главного окна:

6.4.1.1 Главное меню

Главное меню предназначено для управления выполнением программы. Оно доступно на любом этапе выполнения. Команды меню предназначены для быстрого доступа к средствам управления и основным экранным формам программы. Главное меню состоит из следующих разделов «Файл», «Сервис», «Справка».

1. В меню файла можно выполнять действия по открытию и загрузки нового изображения в программу, открывать и просматривать отчеты, а также завершать работу с программой;
2. Меню сервиса предназначено для открытия дополнительных экранных форм программы таких как окно навигатора, окно 3D модели и настроек;
3. В разделе справка можно получить справочную информацию по работе с программой, а также информацию о версии программы и фирме разработчике.

6.4.1.2 Панель инструментов

Панель инструментов предоставляет дополнительные инструменты управления программой. В общем случае содержит группы инструментов для управления выполнением программы, редактирования и масштабирования изображений, а также инструменты для работы со справочной системой программы. Оператору программы доступны следующие инструменты:

 «Открыть» – кнопка загрузки изображения в программу

 «Отчет» – кнопка формирования отчета.

 «Запуск» – кнопка запуска расчета изображения.

 «Регион» – инструмент выделения области стволов бревен.

 «Редактор» – инструмент ручного редактирования бревен.

 «Калибровка» – инструмент калибровки изображения.

 «Справка» – кнопка вызова справочной информации.

 «Увеличить»
 «Уменьшить»
 «Развернуть»
 «Масштаб 100%» – группа инструментов масштабирования изображения. Эти кнопки изменяют масштаб изображения и выделенных бревен внутри рабочей области.

Для выбора конкретного инструмента или команды необходимо щёлкнуть по нему мышкой. При этом на самой панели значок инструмента будет изображен в рамочке, указывая, что в текущий момент активен именно он.

6.4.1.3 Рабочая область

Рабочая область предназначенная для работы с изображениями срезов бревен. В зависимости от активного инструмента рабочая область позволяет оператору осуществлять следующие действия:

– выделять и редактировать калибровочный объект на изображении. Данное действие доступно при выборе инструмента «Калибровка»  ;

– выделять и редактировать отдельные срезы бревен на изображении. Данное действие доступно при выборе инструмента «Редактор»  ;

– выделять произвольную области внутри изображения для ограничения области поиска срезов бревен в автоматическом режиме. Действие доступно при выборе инструмента «Регион»  .

6.4.1.4 Строка состояния

Строка состояния обеспечивает вывод сообщений оператору и результатов работы программы в текстовом и графическом виде. Содержит различную информацию о процессе выполнения программы (см. Рисунок 6.3).

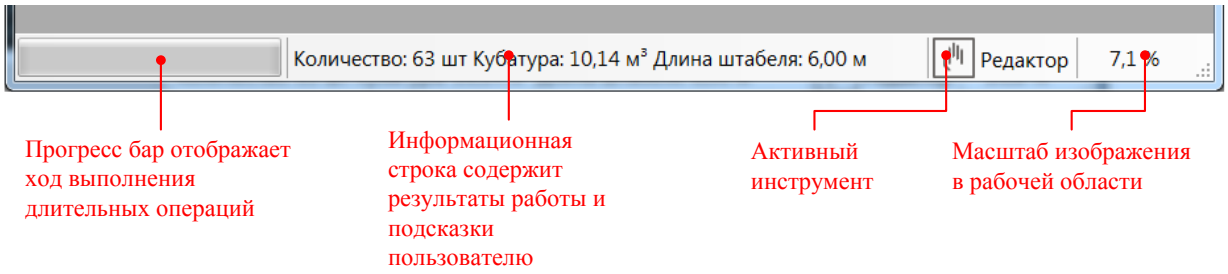


Рисунок 6.3 – Строка состояния

Также строка состояния содержит элемент управления, который визуально показывает ход выполнения длительных фоновых операции программы.

6.4.2 Окно настроек программы

Окно настроек предоставляет пользователю гибкий инструмент по конфигурированию настроек приложения. Все изменения, сделанные пользователем мгновенно отражаются в программе.

Оператор имеет возможность самостоятельно настраивать следующие параметры:

– *стратегия*. Стратегия определяет алгоритм расчета объёма штабеля по одному или двум изображениям.

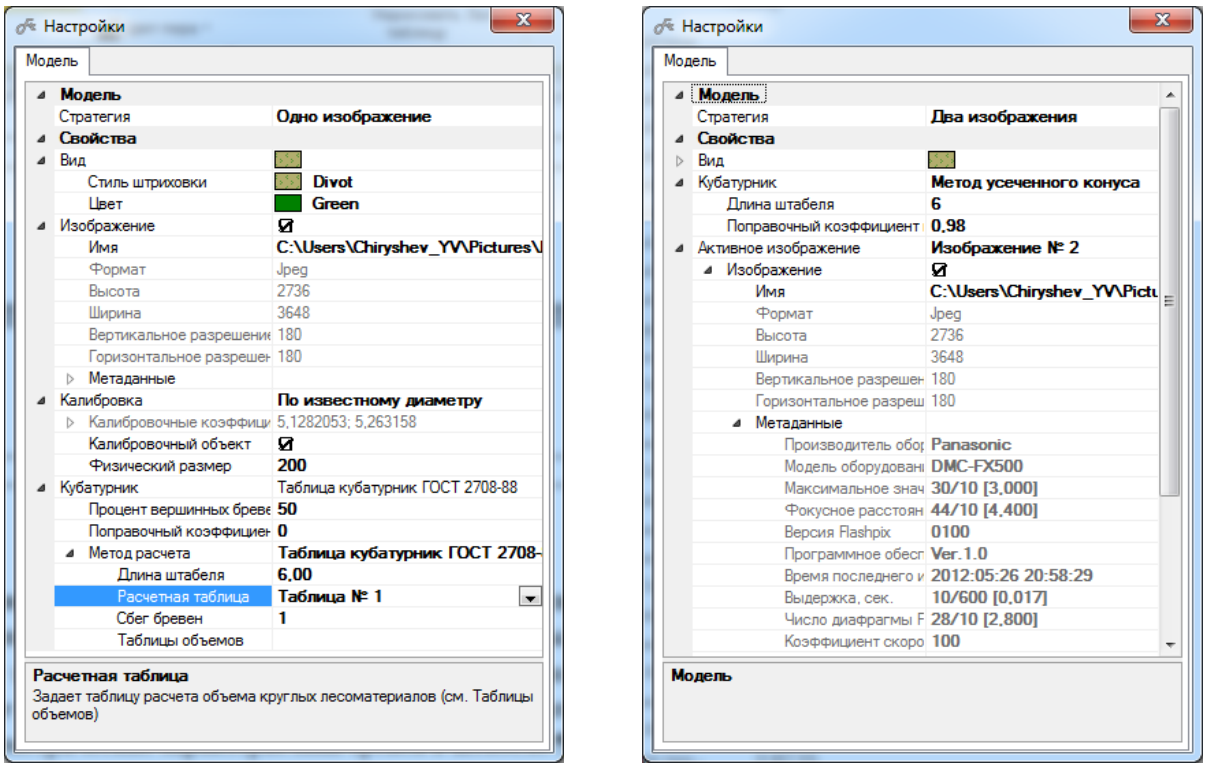
– *внешний вид*. Определяет стиль рисования результатов работы алгоритма в главном окне программы (цвет и текстура изображения срезов бревен).

– *калибровка*. В настройках калибровки имеется возможность выбора метода юстировки и задания его параметров.

– *кубатурник*. В категории определены методы расчета объема согласно действующим отраслевым документам.

– *изображение*. Предоставляет пользователю механизм загрузки изображений в программу и просмотра информации об условиях и способах их получения.

Внешний вид окна настроек показан на Рисунке 6.4.



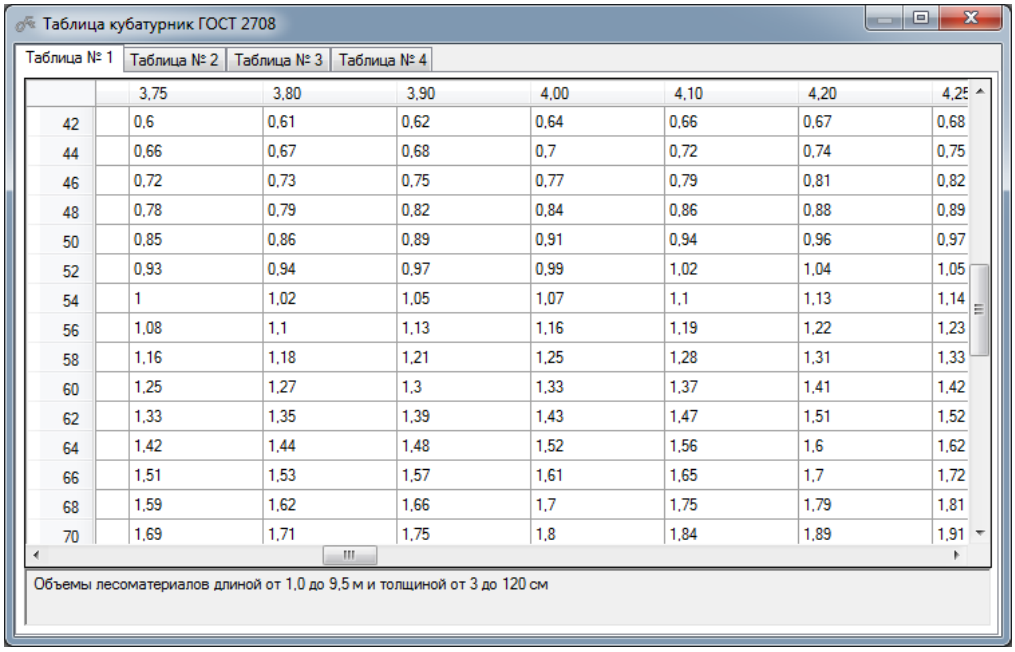
а) б)

Рисунок 6.4 – Внешний вид окон с настройками приложения а) для обработки одного изображения; б) для случая двух изображений

Открытие окна настроек осуществляется из главного окна программы, через меню «Сервис» → «Настройки».

6.4.3 Окно таблиц объемов

В окне приведены объемы круглых лесоматериалов, определяемые по толщине верхнего торца и длине бревна согласно ГОСТ 2708-75. Внешний вид окна таблиц объемов показан на Рисунке 6.5.



	3,75	3,80	3,90	4,00	4,10	4,20	4,25
42	0,6	0,61	0,62	0,64	0,66	0,67	0,68
44	0,66	0,67	0,68	0,7	0,72	0,74	0,75
46	0,72	0,73	0,75	0,77	0,79	0,81	0,82
48	0,78	0,79	0,82	0,84	0,86	0,88	0,89
50	0,85	0,86	0,89	0,91	0,94	0,96	0,97
52	0,93	0,94	0,97	0,99	1,02	1,04	1,05
54	1	1,02	1,05	1,07	1,1	1,13	1,14
56	1,08	1,1	1,13	1,16	1,19	1,22	1,23
58	1,16	1,18	1,21	1,25	1,28	1,31	1,33
60	1,25	1,27	1,3	1,33	1,37	1,41	1,42
62	1,33	1,35	1,39	1,43	1,47	1,51	1,52
64	1,42	1,44	1,48	1,52	1,56	1,6	1,62
66	1,51	1,53	1,57	1,61	1,65	1,7	1,72
68	1,59	1,62	1,66	1,7	1,75	1,79	1,81
70	1,69	1,71	1,75	1,8	1,84	1,89	1,91

Рисунок 6.5 – Внешний вид окна с таблицами объемов

Окно с таблицами объемов доступно для просмотра при выборе соответствующего метода в настройках программы (см. раздел «Окно настроек программы»).

6.4.4 Окно открытия изображений

Окно открытия изображений предназначено для загрузки фалов изображений в программу (см. Рисунок 6.6).

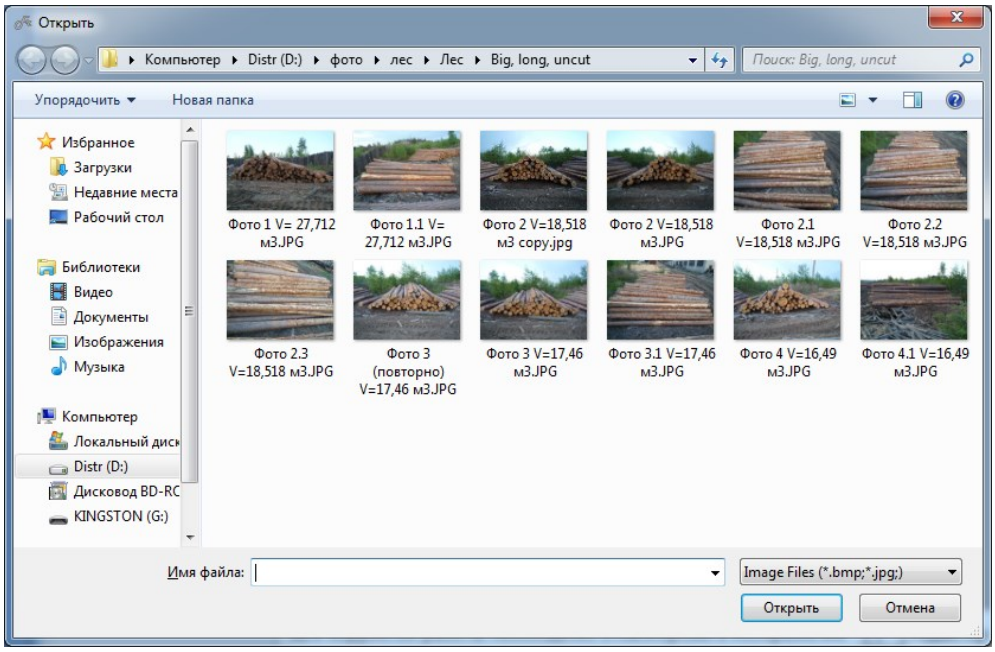


Рисунок 6.6 – Внешний вид окна открытия изображений

Загрузка изображений осуществляется из главного окна программы, через меню «Файл» → «Открыть». Эту возможность также предоставляет специальная кнопка «Открыть» , расположенная на панели инструментов.

6.4.5 Окно навигатора

Окно навигатора предназначено для масштабирования изображений в рабочей области и перемещения по нему (см. Рисунок 3.5).

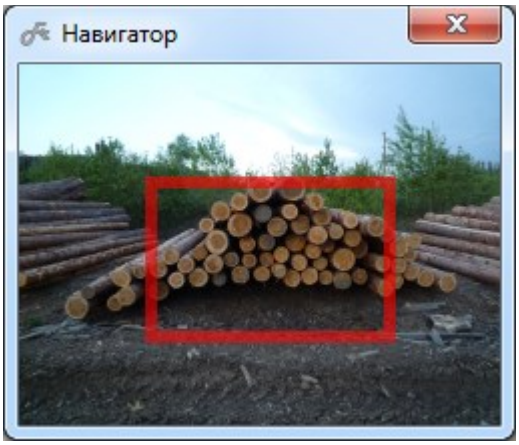


Рисунок 6.7 – Внешний вид навигатора

Основная часть окна занята эскизом изображения, на котором красной рамкой отмечена видимая часть рабочей области. Оператор можем перемещать рамку по эскизу, тем самым «прокручивая» изображение в главном окне программы.

Открытие окна настроек осуществляется из главного окна программы, через меню «Сервис» → «Навигатор».

6.4.6 Окно справочной информации

Окно справки содержит справочную информацию по любым вопросам, которые могут возникнуть в процессе работы с программой (см. Рисунок 6.8).

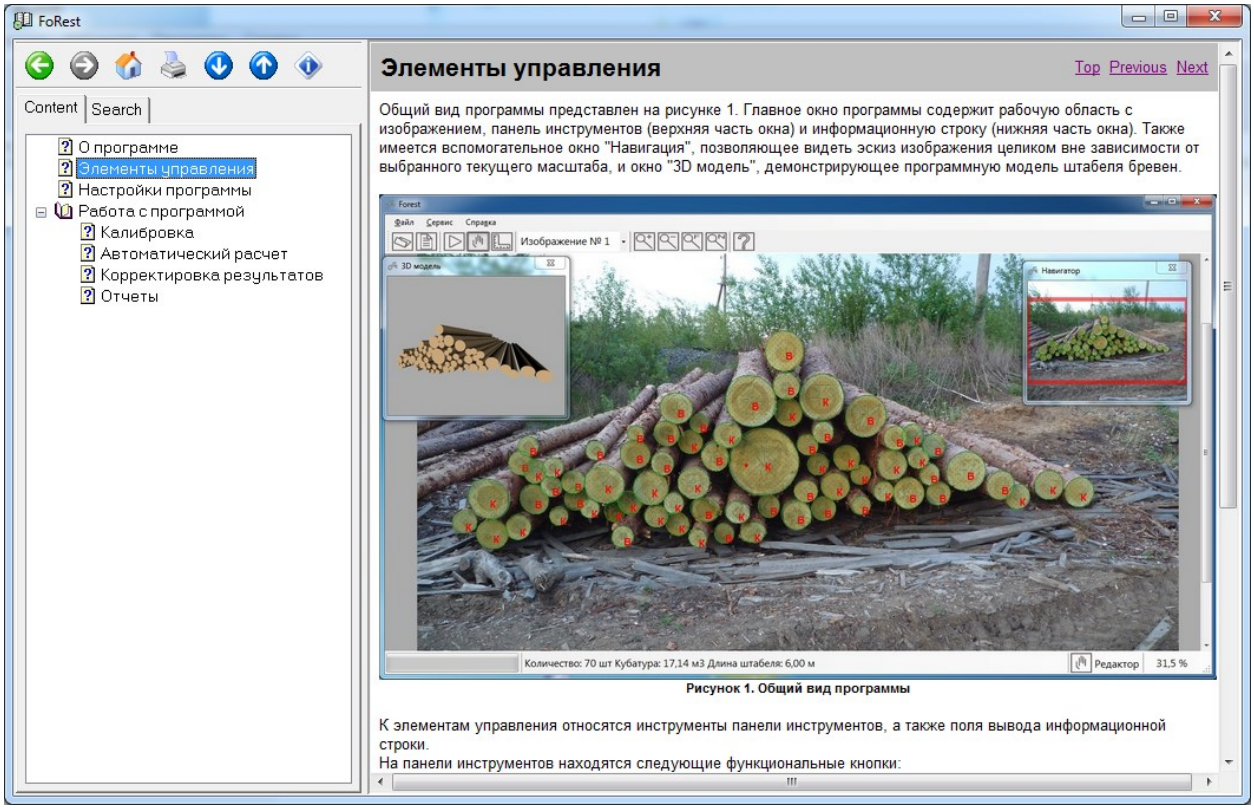
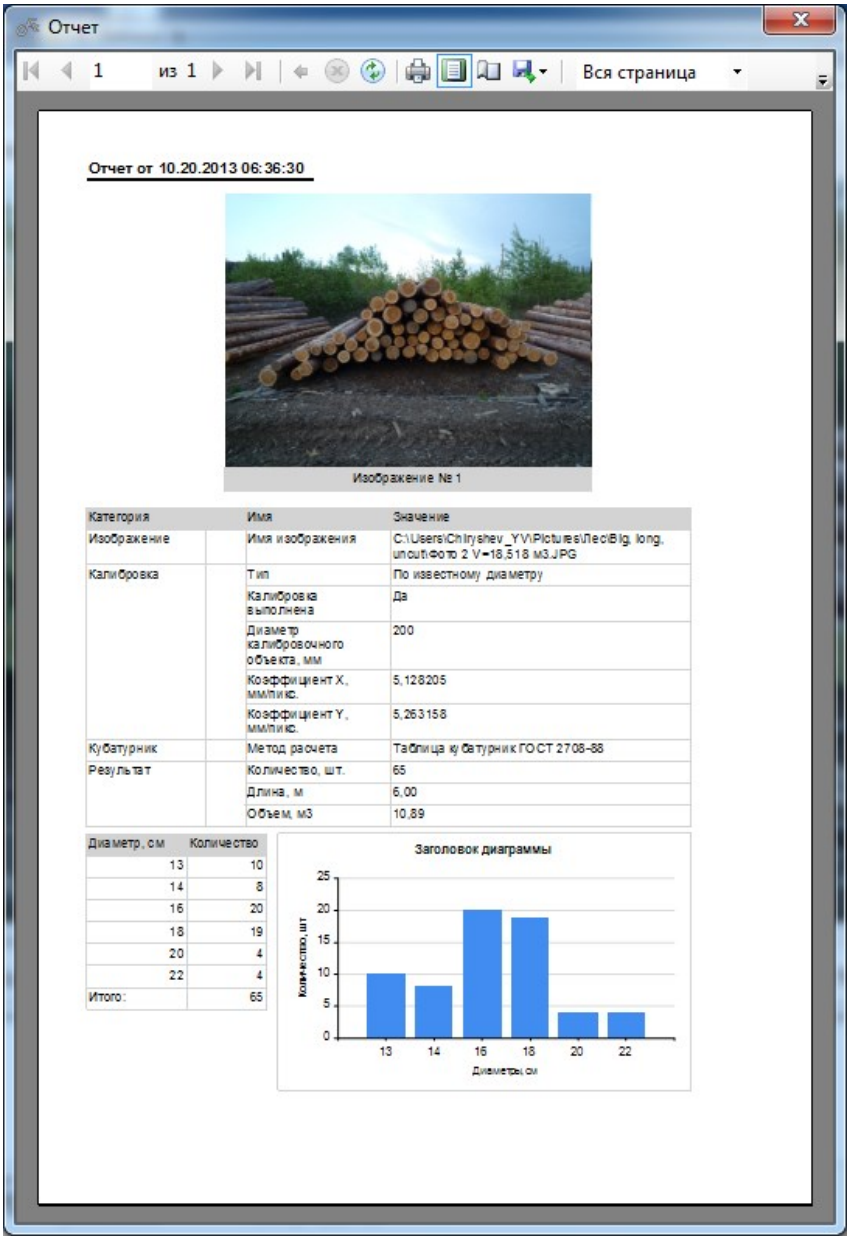


Рисунок 6.8 – Внешний вид окна справки.

Вызов справочника программы осуществляется из главного окна программы через специальную кнопку справки на панели инструментов или через меню «Справка» → «Содержание».

6.4.7 Окно отчетов

Окно отчетов предназначено для просмотра и сохранения результатов работы Программы «FoRest». Внешний вид окна отчетов показан на Рисунке 6.9.



Категория		Имя	Значение
Изображение		Имя изображения	C:\Users\Chiryshev_YV\Pictures\Teci\Big. long, uncut\Фото 2 V=18,518 м3.JPG
Калибровка		Тип	По известному диаметру
		Калибровка выполнена	Да
		Диаметр калибровочного объекта, мм	200
		Коэффициент X, мм/пикс.	5,128205
		Коэффициент Y, мм/пикс.	5,263158
Кубатурник		Метод расчета	Таблица кубатурник ГОСТ 2708-88
Результат		Количество, шт.	65
		Длина, м	6,00
		Объем, м3	10,89

Диаметр, см	Количество
13	10
14	8
16	20
18	19
20	4
22	4
Итого:	65

Рисунок 6.9 – Внешний вид окна с отчетом

Отчет отражает анализируемое изображение, информацию об общем количестве и общем объеме всех бревен, параметрах, при которых получен результат, сводную таблицу, содержащую информацию о распределении бревен по размерам и гистограмму распределения диаметров. Программа позволяет сохранять отчеты в одном из наиболее популярных форматов: DOC, XLS или PDF.

Открытие окна отчетов осуществляется из главного окна программы, через специальную кнопку отчета на панели инструментов или последовательным выбором пунктов меню «Файл» → «Отчет».

6.5 Выполнение программы

6.5.1 Установка программы

Полная установка Программы «FoRest» предполагает инсталляцию файлов программы на персональный компьютер пользователя, а также необходимых для ее запуска и работы дополнительных программных пакетов (см. раздел 2.2 «Минимальный состав программных средств»). Допускается установка только одной копии программы.

Шаг 1. Запуск установки. Для начала процесса установки программы на персональный компьютер необходимо запустить исполняемый файл *setup.exe* с установочного диска.

Шаг 2. Установка программных компонент. Для реализации всех возможностей программы «FoRest» перед установкой пользователю будет предложено установить необходимые программные компоненты (см. раздел «Минимальный состав программных средств»). Для успешной установки программы необходимо согласиться с установкой всех требуемых элементов (см. Рисунок 6.10). Установка программных компонентов будет происходить автоматически.

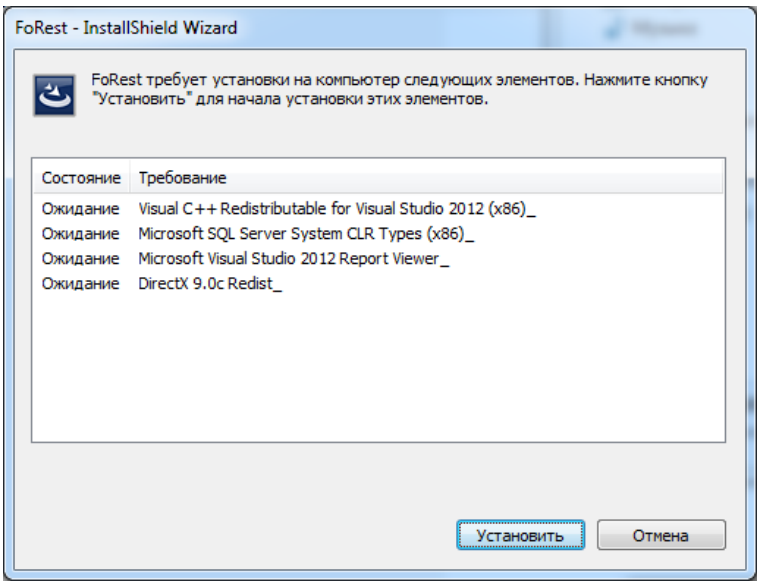


Рисунок 6.10 – Установка программных компонент.

Шаг 3. Установка программы. После установки всех необходимых пакетов программа установки предложит пользователю установить Программу «FoRest». Для продолжения установки необходимо выбрать команду **Далее** или отменить установку (см. Рисунок 6.11).

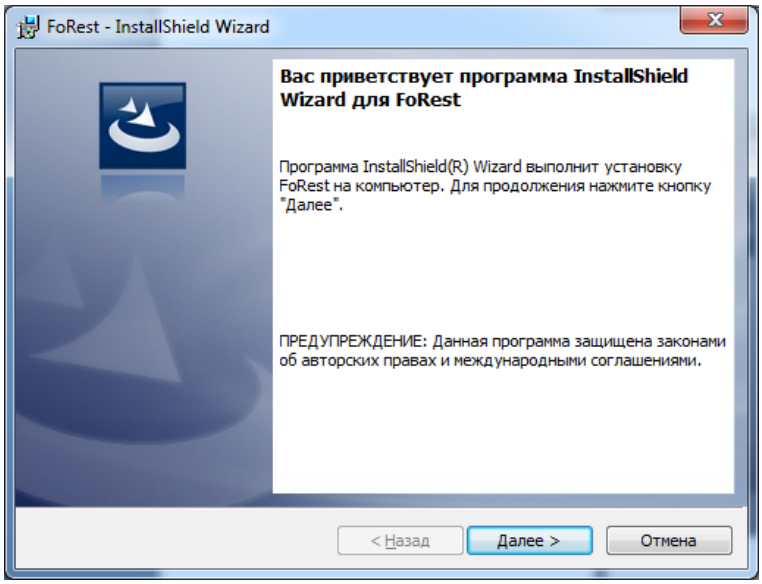


Рисунок 6.11 – Начало установки

Шаг 4. Подтверждение установки. В окне необходимо указать действие, которое будет производить программа инсталляции (см. Рисунок 6.12). Это может быть **Установить**, **Назад** или **Отмена**. Для подтверждения установки программы необходимо выбрать первую команду.

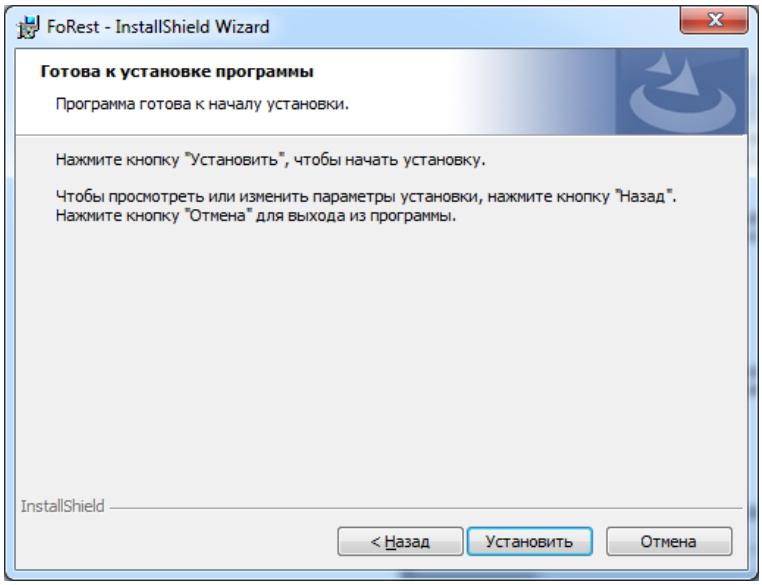


Рисунок 6.12 – Подтверждение установки

Шаг 5.Установка. При запуске установки выводится окно, которое извещает о необходимости подождать, пока программные файлы будут установлены на персональный компьютер (см. Рисунок 6.13). Установка программы проходит автоматически и не требует вмешательства пользователя.

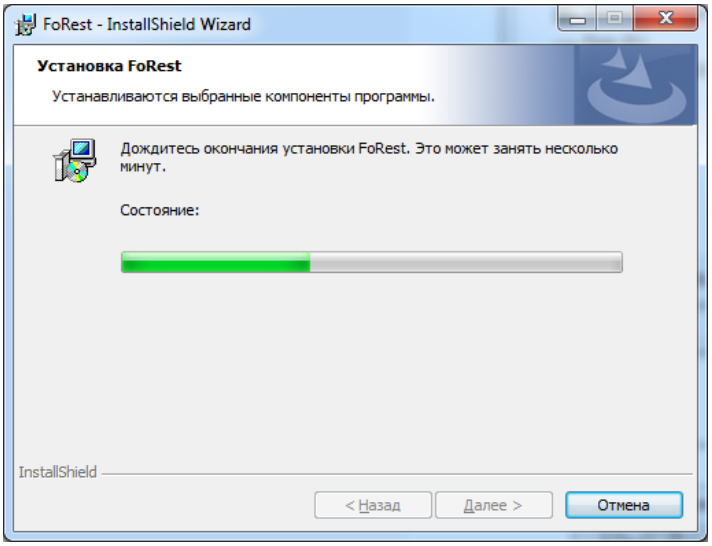


Рисунок 6.13 – Отображение процесса установки программы

После успешной установки программы пользователю будет выведено соответствующее окно (см. Рисунок 6.14). Для выхода из программы установки необходимо нажать кнопку **Готово.**

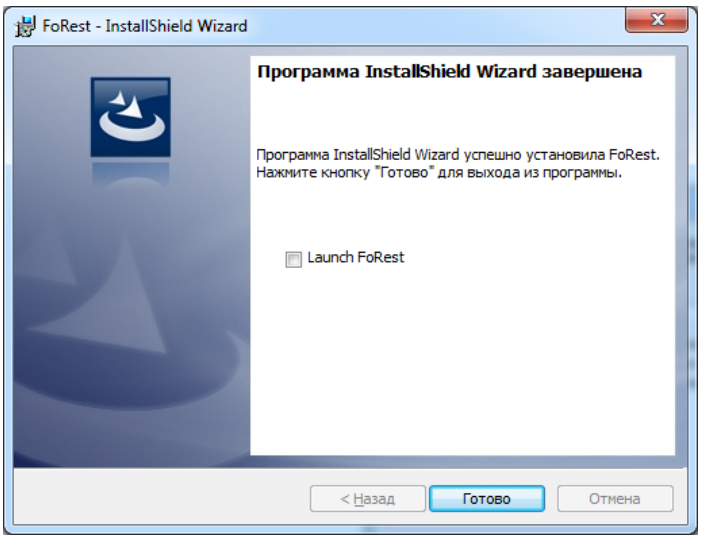


Рисунок 6.14 – Завершение установки программы

Шаг 6. Регистрация и активизация программы. В зависимости от версии программы, после установки при запуске пользователю будет

предложено зарегистрировать и активизировать программу. Альтернатива этому – возможность в течение 30 дней использовать программу без регистрации для знакомства с ее возможностями.

6.5.2 Запуск и выполнение программы

Запуск программы осуществляется путем активации файла приложения *Forest.exe* из папки установки (по умолчанию это *C:\Program Files\Kvinta\FoRest*) или ярлыка на Рабочем столе.

Выполнение программы для получения результатов измерения штабеля круглого леса определяется следующей последовательностью действий:

1. загрузка изображений;
2. калибровка изображений;
3. автоматический поиск бревен;
4. ручное редактирование;
5. получение и анализ результата.

6.5.3 Загрузка изображений в программу

Загрузка изображений осуществляется из главного окна программы, через меню «Файл» → «Открыть». Эту возможность также предоставляет специальная кнопка «Открыть» , расположенная на панели инструментов.

Перед началом работы оператору необходимо выбрать количество изображений штабеля круглого леса, для которого будет производится процедура измерения. Допустимое количество изображений задается в настройках программы в разделе «Модель» →«Стратегия» (см. Рисунок 6.15).

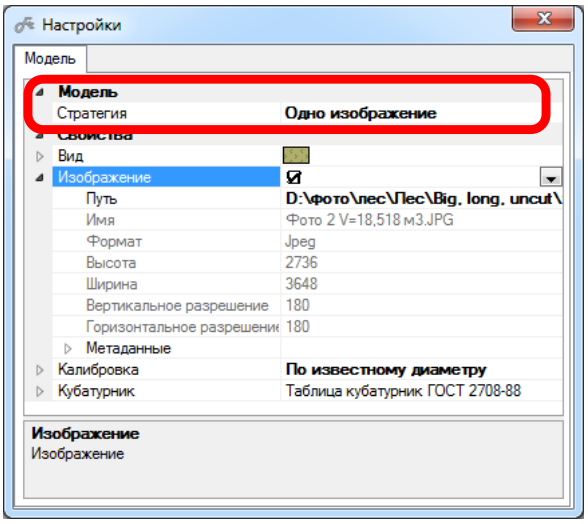


Рисунок 6.15 – Раздел «Стратегия» в настройках программы

На Рисунке 6.16 приведено главное окно программы после загрузки изображения с бревнами.

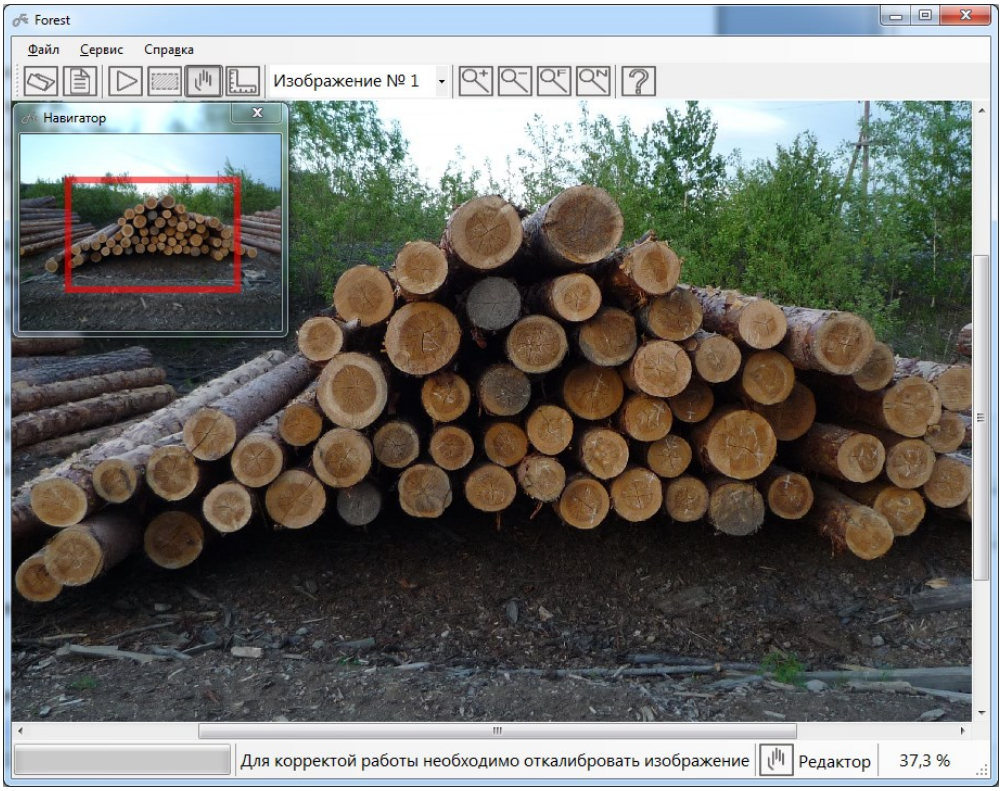


Рисунок 6.16 – Внешний вид главного окна программы с загруженным изображением

В программу допускается загружать как цветные, так и полутоновые изображения в сжатом (JPG, JPEG) или несжатом формате (BMP).

На изображения накладываются следующие ограничения:

– минимальное разрешение загружаемого файла определяется масштабом и количеством бревен на изображении. Не допускается загружать изображения, на которых торцы искомых бревен меньше 5 пикселей. Нарушение данного требования может привести к неправильному выделению торцов бревен;

– максимальный для загрузки размер изображений определяется объемом оперативной памяти, но не может быть менее 800x600 пикселей. Нарушение данного требования может привести к неправильному выделению торцов бревен и измерению объема штабеля.

6.5.4 Калибровка изображений

Запуск процедуры калибровки осуществляется активацией инструмента «Калибровка»  на панели инструментов (см. раздел «Главное окно программы»).

Для проведения калибровки используется метод сопоставления с эталоном. Калибровочным объектом является одно из бревен пакета, размеры которого должны быть заранее измерены ручным способом перед расчетом изображения программными средствами. От оператора программы требуется:

– *указать интересующее бревно*. Выделение эталонного бревна осуществляется рисованием на изображении эллипса вокруг среза при помощи манипулятора мыши последовательно указав три точки его границы: две точки главной оси и третья точка для построения эллипса (см. Рисунок 6.23);

– *задать для бревна физический размер*. Диаметр калибровочного объекта задается в миллиметрах в окне настроек в разделе «Калибровка» → «Физический размер» (см. Рисунок 6.17).

Программа автоматически определит точные границы объекта и рассчитает результат. В случае неточного выделения контура бревна в

автоматическом режиме можно воспользоваться ручной коррекции размера калибровочного объекта.

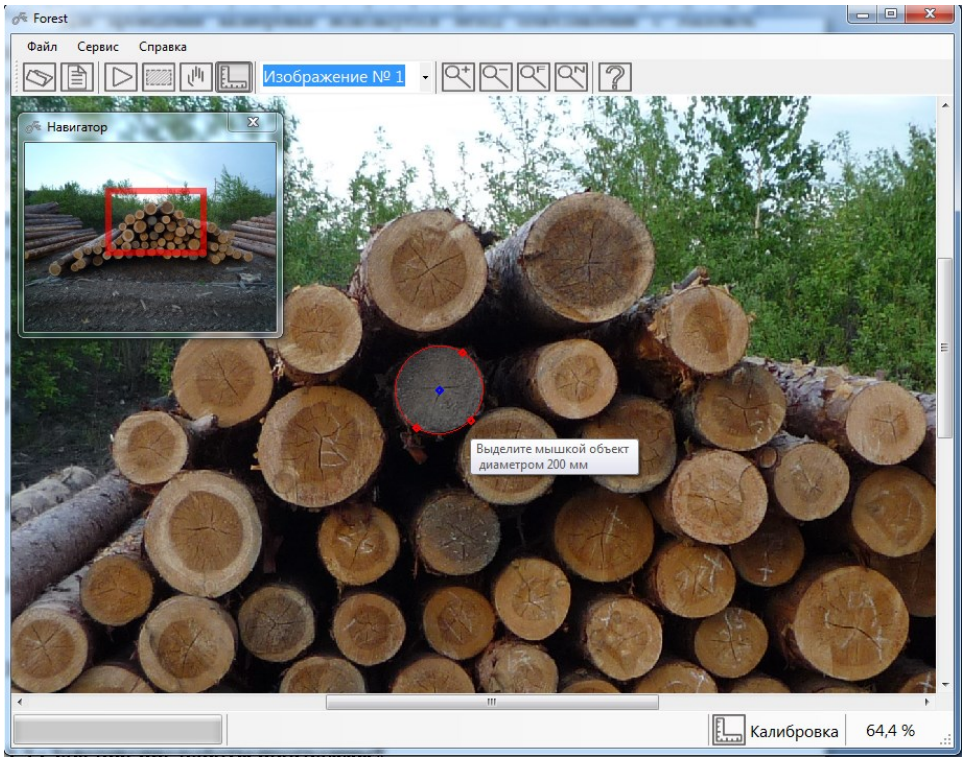


Рисунок 6.17 – Выделение эталонного бревна

Результатом калибровки являются два числа, определяющие цену условной единицы пикселя изображения по вертикали и горизонтали. Результаты калибровки автоматически заносятся в программу и доступны для просмотра в окне настроек в разделе «Калибровка» → «Калибровочные коэффициенты» (см. Рисунок 6.18)

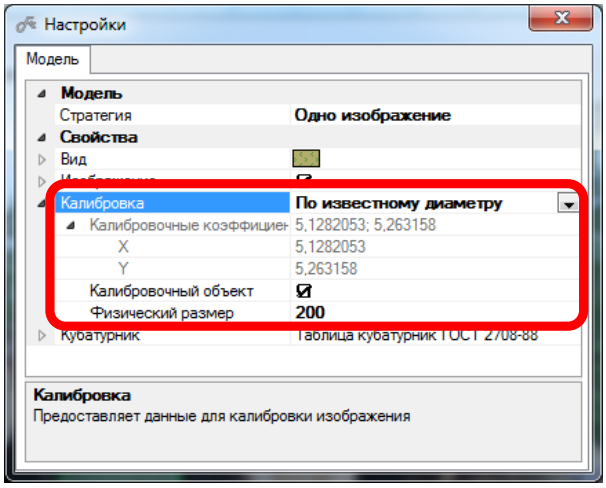


Рисунок 6.18 – Раздел «Калибровка» в настройках программы

6.5.5 Автоматический поиск бревен

После выполнения калибровки может быть выполнен поиск бревен на изображении. Алгоритм автоматического поиска запускается соответствующей кнопкой  на панели инструментов. При этом индикатор в строке состояния показывает, что идет процесс обработки (см. Рисунок 6.19).

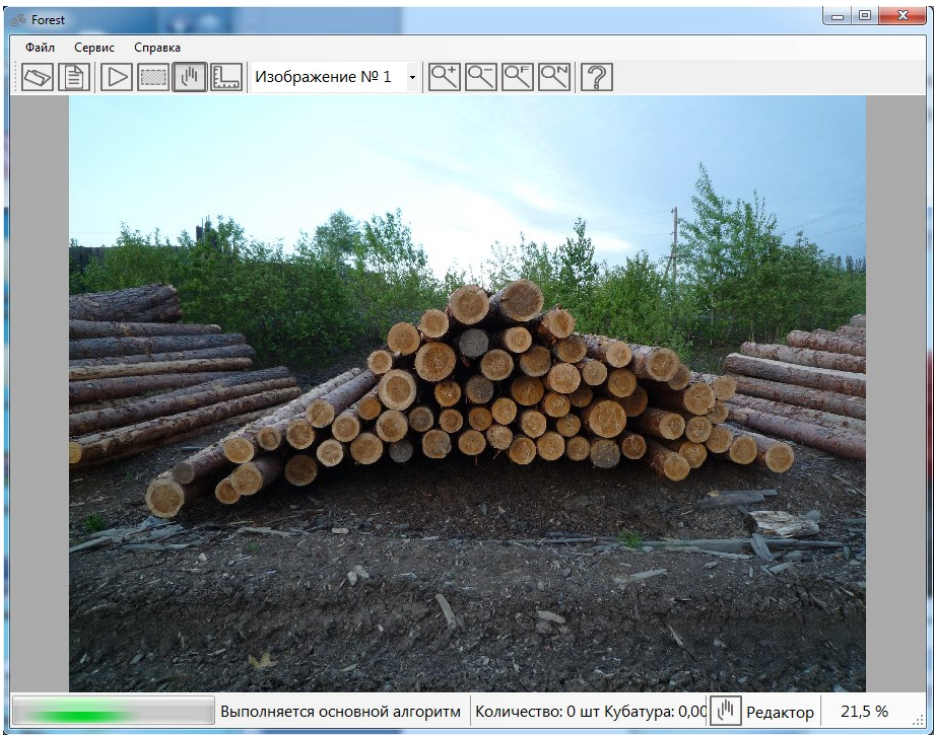


Рисунок 6.19 – Внешний вид главного окна при запуске автоматического поиска бревен

По завершении автоматического поиска все найденные торцы бревен будут выделены цветным маркером (по умолчанию цвет маркера – зеленый) и определена ориентация каждого бревна «Комель» или «Вершина»:

– В случае обработки по двум изображениям ориентация бревен определяется путем сопоставления двух торцов каждого бревна и присваивания большему торцу значения «Комель», а меньшему – «Вершина».

– В случае обработки по одному изображению ориентация назначается выборочно на основе анализа размеров торцов всех бревен в пакете.

На изображении ориентация показывается буквами «К» или «В» на торце каждого бревна для комля или вершины соответственно. Результат работы измерения по каждому отдельному бревну доступен пользователю через всплывающее окно при наведении манипулятора мыши на соответствующий торец и включает следующие характеристики:

– метод расчета;
– ориентация видимого торца бревна, комель/вершина;
– верхний и нижний диаметры, см;
– длина бревна, м;
– объем, $м^3$.

В информационную строку выводятся результаты анализа изображения – общее количество и объем бревен (см. Рисунок 6.20).

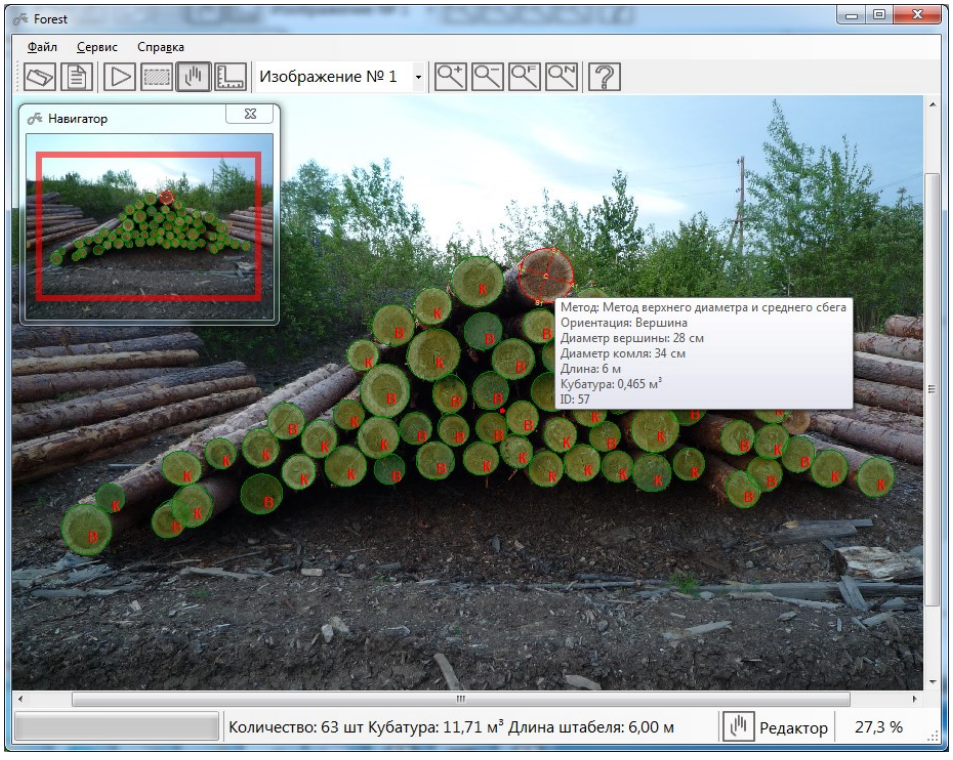


Рисунок 6.20 – Результат работы автоматического выделения бревен

Расчет объема осуществляется в соответствии с действующими нормативными документами (ГОСТ 32594-2013 [79] и ГОСТ 2708-75 [80]). Метод расчета задается в настройках программы в разделе «Кубатурник» →«Метод расчета» (см. Рисунок 6.21).

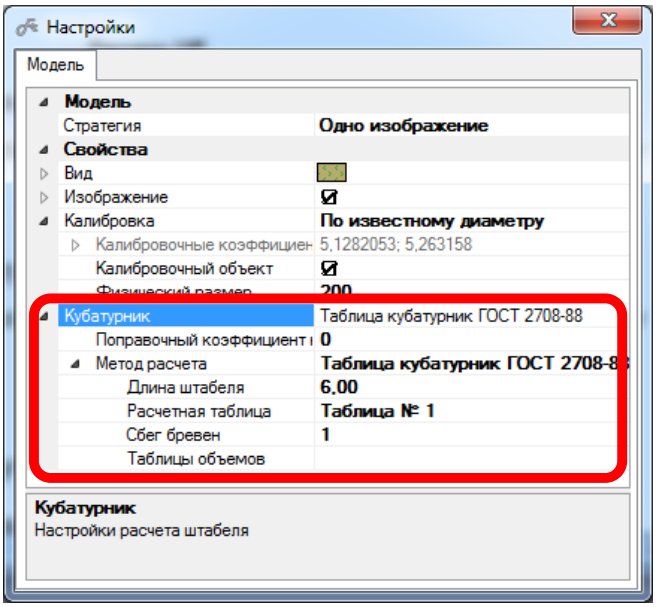


Рисунок 6.21 – Раздел «Кубатурник» в настройках программы

Каждый метод в разделе кубатурника содержит дополнительные параметры доступные для настройки оператором программы:

– поправочный коэффициент на объем коры;

– длина пакета бревен;

– сбег бревен;

– расчетная таблица.

После настройки метода и параметров расчета программа автоматически пересчитает результат с учетом сделанных изменений.

Перед запуском анализа изображения оператор программы может выделить произвольную область внутри изображения для ограничения области поиска срезов бревен (см. Рисунок 6.22). Данное действие доступно при выборе инструмента «Регион» в главном окне программы.

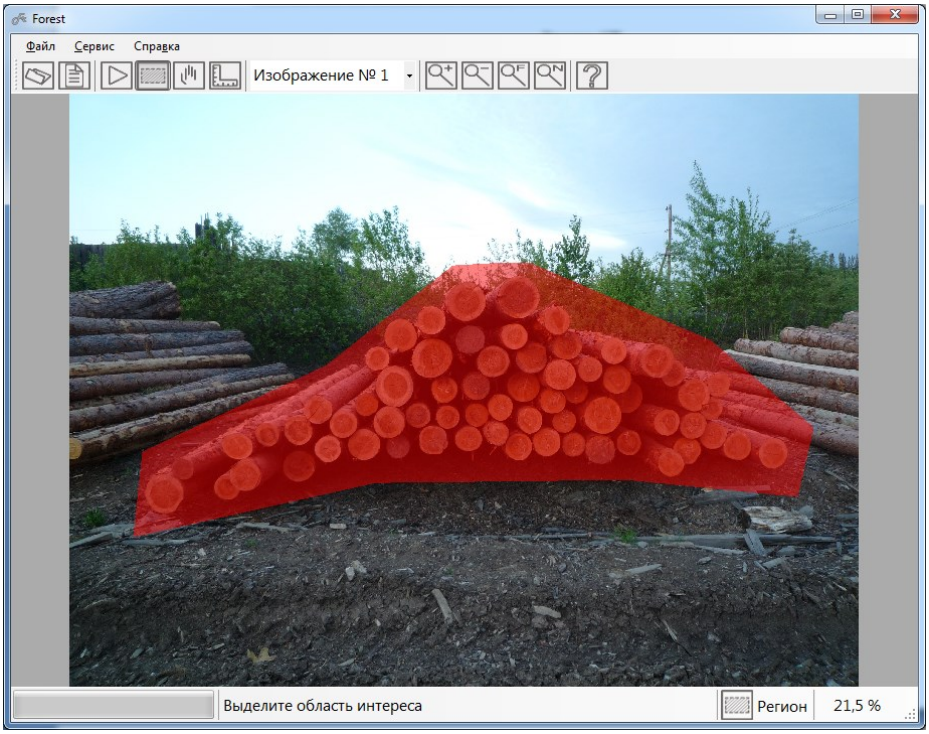


Рисунок 6.22 – Область поиска, выделенная внутри изображения

Автоматический поиск торцов бревен можно выполнить до этапа калибровки, однако в этом случае объем штабеля рассчитать будет невозможно и программа выдаст предупреждение (см. раздел «Сообщения оператору»).

6.5.6 Ручное редактирование

В результате выполнения автоматического расчета часть торцов может оказаться невыделенной, и при этом часть объектов может быть выделена ошибочно. Для подобных ситуаций в программе предусмотрен механизм ручного редактирования. Действия по редактированию доступны при выборе соответствующего инструмента «Редактор» на панели инструментов в главном окне программы.

Оператору доступны следующие действия по редактированию:

– *удаление торцов*. Торцы могут быть удалены путем их выделения и выбора в контекстном меню пункта «Удалить»;

– *добавление торцов*. Для добавления бревен используется тот же принцип, что и при калибровке – рисование эллипса по трем точкам (см. раздел 3.2.2 «Калибровка изображений»);

– *изменение размеров и положения торцов*. При ручной корректировке можно изменить эти параметры: при выделении щелчком правой кнопкой мыши торца становятся доступны области, за которые его можно перемещать, растягивать и вращать.

– *изменение ориентации.* Основным параметром торца среди прочих является его ориентация (комель–вершина). Изменить автоматически назначенную ориентацию можно в соответствующей вкладке контекстного меню, как показано на Рисунке 6.23.

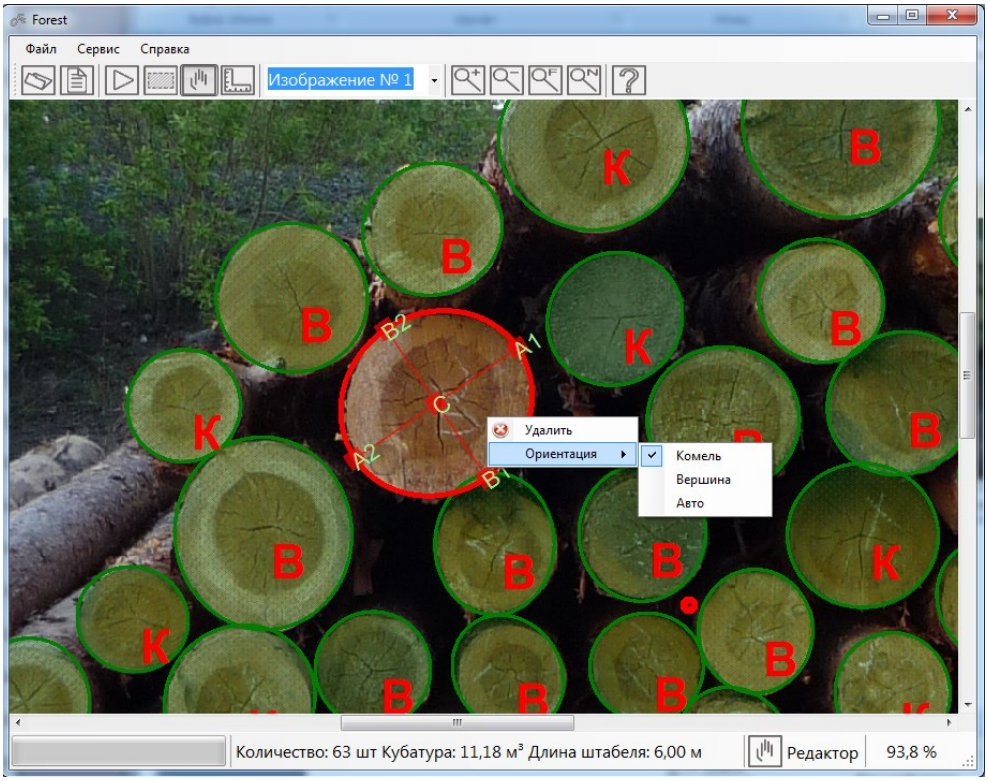


Рисунок 6.23 – Ручное редактирование

После выполнения ручного редактирования программа автоматически пересчитает результат с учетом сделанных изменений.

6.5.7 Получение и анализ результата

Для анализа результатов работы программы предусмотрен модуль детализированных отчетов (см. раздел «Окно отчетов»).

6.5.8 Завершение работы программы

Выполнение указанной функции возможно любым из перечисленных ниже способов:

1. Последовательным выбором пунктов меню «Файл» → «Выход»;
2. В верхнем правом углу главного окна кликнуть левой кнопкой мыши по кнопке в виде крестика **Закрыть**.

6.6 Сообщения оператору

Все служебные сообщения выводятся оператору в строке состояния или рабочей области при наведении на нее манипулятора мыши. Текст сообщений, выдаваемых в ходе выполнения программы, и описание их содержания приведены в Таблице 6.1.

Таблица 6.1 – Сообщения оператору

Текст сообщения	Описание
Для корректной работы необходимо откалибровать изображение	Сообщение выводится в строке состояния и указывает на необходимость выполнения процедуры калибровки для изображения (см. раздел «Калибровка изображений»). Данное сообщение выводится в случае измерения штабеля по одному изображению.
Для корректной работы необходимо откалибровать оба изображения	Сообщение выводится в строке состояния и указывает на необходимость выполнения процедуры калибровки для обоих изображений (см. раздел «Калибровка изображений»). Данное сообщение выводится в случае измерения штабеля по двум изображениям.

Текст сообщения	Описание
Изображение неоткалибровано	Сообщение выводится как всплывающая подсказка в рабочей области главного окна программы. Данное сообщение выводится для каждого бревна, при наведении на него манипулятора мыши и указывает на необходимость выполнения процедуры калибровки (см. раздел «Калибровка изображений»), при этом бревно отображается с предупреждающим знаком 
Выделите мышкой объект диаметром N мм.	Сообщение выводится как всплывающая подсказка в рабочей области главного окна программы. Сообщение предлагает пользователю выделить эталонный объект диаметром N миллиметров при выполнении процедуры калибровки. N – это физический размер объекта, указанный в настройках программы.
Выделите область интереса	Сообщение выводится в строке состояния главного окна программы и информирует пользователя о необходимости выделить область поиска срезов бревен Данное сообщение доступно при выборе инструмента «Регион»  в главном окне программы.
Ошибка расчетов. Проверти правильность данных.	Сообщение выводится как всплывающая подсказка в рабочей области главного окна программы. Данное сообщение доступно для бревна при наведении на него манипулятора мыши, при этом срез бревна отображается с предупреждающим знаком  . Пользователю необходимо проверить правильность введенных данных в настройках программы в разделах «Калибровка» и «Кубатурник»

Текст сообщения	Описание
Ошибка ключа столбца (длина пакета, м) Данный ключ отсутствует в словаре	Сообщение выводится как всплывающая подсказка в рабочей области главного окна программы. Данное сообщение доступно для бревна при наведении на него манипулятора мыши, при этом срез бревна отображается с предупреждающим знаком ⚠. Пользователю необходимо проверить правильность введенных данных в настройках программы в разделе «Кубатурник». Как правило это сообщение выводится, если методом расчета является **Таблица ГОСТ 2708** и вместе с этим указана длина бревен, отсутствующая в используемой таблице.
Ошибка ключа строки (диаметр, см) Данный ключ отсутствует в словаре	Сообщение выводится как всплывающая подсказка в рабочей области главного окна программы. Данное сообщение доступно для бревна при наведении на него манипулятора мыши, при этом срез бревна отображается с предупреждающим знаком ⚠. Пользователю необходимо проверить правильность введенных данных в настройках программы в разделе «Кубатурник». Как правило это сообщение выводится, если методом расчета является **Таблица ГОСТ 2708**.
Верхний диаметр имеет недопустимое значение	Сообщение выводится как всплывающая подсказка в рабочей области главного окна программы. Данное сообщение доступно для бревна при наведении на него манипулятора мыши, при этом срез бревна отображается с предупреждающим знаком ⚠. Пользователю необходимо проверить правильность введенных данных в настройках программы в разделах «Калибровка» и «Кубатурник». Как правило сообщение выводится, если указан большой сбег бревен.

6.7 Выводы

Основная задача, решаемая программой – это измерение объема и геометрических характеристик уложенного в штабели круглого лесоматериала посредствам обработки изображений торцов бревен специализированными методами машинного зрения и предоставление результатов измерения оператору программы в виде отчета.

Программное обеспечение состоит из следующих частей:

– Основная исполняемая программа Forest.exe. Включает пользовательский интерфейс и методы цифровой обработки изображений для поиска срезов бревен на изображении;

– Вспомогательная программа ForestHelp.exe. Вызывается основной программой и содержит справочную информацию по работе с программой;

– Библиотека ForestCubaturnic.dll. Реализует поштучный метод расчета объема штабеля в соответствии с действующими нормативными документами;

– Библиотека ForestReporting.dll. Включает методы и классы для просмотра отчетов, созданных с применением технологии отчетности Microsoft.

Все программные компоненты реализованы на языке высокого уровня C++/CLI. Работа с программой предполагает совершения определенной последовательности действий для получения требуемого результата. Схема последовательности действий по работе с основной программой включает следующие этапы.

1) загрузка изображений;
2) калибровка изображений;
3) автоматический поиск бревен;
4) ручное редактирование;
5) измерение объема бревен;
6) получение и анализ результата.

ГЛАВА 7. РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ МЕТОДИКИ МОБИЛЬНОГО РАСЧЕТА ОБЪЕМА ШТАБЕЛЯ КРУГЛОГО ЛЕСА

7.1 Программа и методика исследовательских испытаний программы для мобильного расчета объема штабеля круглого леса

7.1.1 Требования к средствам проведения испытаний

Испытания Программы проводятся в условиях и с использованием следующего оборудования (Таблица 7.1)

Таблица 7.1 – Спецификация оборудования для проведения испытаний

Комплектующие
Планшетный компьютер ASUS Nexus7 4 Core Snapdragon S4
Планшетный компьютер Samsung Galaxy Tab 3 GT0P5210 16 Gb 10,1”

Измерения проводятся в соответствии с методиками ГОСТ 32594-2013 «Лесоматериалы круглые. Методы измерений».

Для обеспечения единства оценки степени тяжести возможных неполадок возникающих при проведении испытаний, вводятся следующие оценки степени тяжести неполадок.

Таблица 7.2 – Оценка степени тяжести неполадок

Степень тяжести неполадки	Условный вес тяжести неполадки	Определение
Критическая	2	Неполадка повлияла на фактическую эффективность работоспособности таким образом, что все зафиксированные фактические значения количественных и качественных характеристик находятся за определенными допустимыми пределами.
Серьезная	1	Неполадка повлияла на фактическую эффективность работоспособности таким образом, что часть зафиксированных фактических значений количественных и качественных характеристик находятся за определенными допустимыми

Степень тяжести неполадки	Условный вес тяжести неполадки	Определение
		пределами.
Незначительная	1	Неполадка повлияла на фактическую эффективность работоспособности таким образом, что некоторые зафиксированные фактические значения количественных и качественных характеристик имеют близкие и незначительные отклонения определенных допустимыми пределами.

7.1.2 Требования к условиям проведения испытаний

Технические средства Программы и персонал должны размещаться в существующих помещениях Исполнителя, которые по климатическим условиям должны соответствовать ГОСТ 15150-69 «Машины, приборы и другие технические изделия. Исполнения для различных климатических районов. Категории, условия эксплуатации, хранения и транспортирования в части воздействия климатических факторов внешней среды» (температура окружающего воздуха от 5 до 40 °С, относительная влажность от 40 до 80 % при Т=25 °С, атмосферное давление от 630 до 800 мм ртутного столба). Размещение технических средств и организация автоматизированных рабочих мест должны быть выполнены в соответствии с требованиями ГОСТ 21958-76 «Система «Человек-машина». Зал и кабины операторов. Взаимное расположение рабочих мест. Общие эргономические требования».

7.1.3 Требования к подготовке изделия к испытаниям

Подготовка к испытаниям заключается в проведении пуско-наладочных работ необходимых программных средств и обучении персонала работе с Программой.

7.1.4 Требования к персоналу, осуществляющему подготовку к испытаниям и испытания

К проведению испытаний допускается персонал, прошедший обучение, изучивший техническую документацию на Программу.

В обязанности представителей Исполнителя входят функции технического руководства при проведении испытаний, измерения количественных показателей работы Программы и анализа результатов испытаний.

Требования к наличию специальных допусков у персонала, проводящего испытания, не предъявляются.

7.1.5 Требования безопасности

При подготовке, проведении испытаний и выполнении работ по завершению испытаний. Исполнитель должен обеспечить соблюдение требований безопасности, установленных:

– ГОСТ 12.2.007.0–75 «Система стандартов безопасности труда. Изделия электротехнические. Общие требования безопасности»;

– «Правилами техники безопасности при эксплуатации электроустановок потребителей»;

– «Правилами технической эксплуатации электроустановок потребителей».

7.1.6 Программа испытаний

Таблица 7.3 – Перечень проверок

Пункт программы	Наименование проверки	Пункт методики
1	Проведение исследовательских испытаний программы для мобильной оценки объема штабеля круглого леса на предмет работоспособности, а также эффективности и удобства пользовательского интерфейса	6.1.8.1

Пункт программы	Наименование проверки	Пункт методики
2	Проведение исследовательских испытаний программы для мобильной оценки объема штабеля круглого леса на предмет появления ошибок первого рода при работе алгоритма автоматического выделения торцов бревен	6.1.8.2.
3	Проведение исследовательских испытаний программы для мобильной оценки объема штабеля круглого леса на предмет появления ошибок второго рода при работе алгоритма автоматического выделения торцов бревен	6.1.8.3.
4	Проведение исследовательских испытаний программы для мобильной оценки объема штабеля круглого леса на предмет соответствия расчетных значений объема фактическим показателям, получаемым посредством ручного поштучного измерения бревен согласно ГОСТ 2708	6.1.8.4.

7.1.7 Режимы испытаний

7.1.7.1 Порядок испытаний

Для проведения испытаний приказом руководителя организации-исполнителя назначается ответственное лицо – руководитель испытаний.

Испытания проводятся в соответствии с планом-графиком, утверждаемым руководителем организации-исполнителя.

Последовательность проведения испытаний может быть изменена по решению руководителя испытаний.

7.1.7.2 Ограничения и другие указания, которые необходимо выполнять на всех или на отдельных режимах испытаний

Испытания прекращаются в случаях:

– явного несоответствия получаемых результатов заявленным характеристикам;

– возникновения аварийных ситуаций;

7.1.7.3 Условия перерыва, аннулирования и возобновления испытаний на всех или на отдельных режимах.

Необходимость, условия и порядок перерыва, аннулирования или прекращения испытаний определяется руководителем испытаний.

7.1.8 Методы испытаний

7.1.8.1 Проверка по п. 1. Программы

Проверка работоспособности Программы в условиях промышленного производства.

Испытание осуществляется на нескольких опорных пунктах лесоперерабатывающего предприятия: на месте заготовки леса, в пункте приемки-отправки леса, на нижнем складе. Общая продолжительность эксплуатации Программы составляет не менее 500 чел/часов. В ходе испытаний должна быть проведена проверка на наличие ошибок (Таблица 7.4).

Таблица 7.4 – Характеристика ошибок

№ п/п	Уровень критичности	Описание
1	Критическая ошибка	Становится полностью или частично невозможна работа Программы (недоступность базы данных, отказ отдельных модулей, крах Программы) или Программа работает с нарушениями
2	Некритическая ошибка	Выявлены недочеты, не влияющие на работу Программы в целом, но требующие исправления
3	Рекомендации	Пожелания по изменению работы отдельных модулей или узлов программы, добавлению функционала

Программа считается выдержавшей испытание, если в ходе испытаний не возникло критических ошибок.

7.1.8.2 Проверка по п. 2 Программы

Испытания программы для мобильной оценки объема штабеля круглого леса на предмет появления ошибок первого рода при работе алгоритма автоматического выделения торцов бревен.

Проводится съемка не менее 400 различных партий леса:

- в штабелированном виде и на лесовозе;
- ориентированными к фотообъективу комлями, торцами и враскомлевку;
- в различных погодных условиях (снег, солнечная погода).

На каждом типе оборудования (Таблица 7.1) должно быть отснято не менее 150 кадров.

Условия съемки: камера расположена параллельно плоскости торцов пакета бревен, пакет расположен по центру кадра и имеются отступы от крайних бревен пакета до границы кадра

Общее количество измерений – не менее 600.

Программа считается выдержавшей испытание, если процент ошибок 1-го рода (ложное срабатывание) в результате работы алгоритма автоматического выдеения торцов бревен не превышает 5% от общего числа выделенных объектов на изображении для каждого изображения.

7.1.8.3 Проверка по п. 3 Программы

Испытания программы для мобильной оценки объема штабеля круглого леса на предмет появления ошибок второго рода при работе алгоритма автоматического выделения торцов бревен.

Проводится съемка не менее 400 различных партий леса:

- в штабелированном виде и на лесовозе;
- ориентированными к фотообъективу комлями, торцами и враскомлевку;
- в различных погодных условиях (снег, солнечная погода).

На каждом типе оборудования (Таблица 7.1) должно быть отснято не менее 150 кадров.

Условия съемки: камера расположена параллельно плоскости торцов пакета бревен, пакет расположен по центру кадра и имеются отступы от крайних бревен пакета до границы кадра

Общее количество измерений – не менее 600.

Программа считается выдержавшей испытание, если процент ошибок 2-го рода (пропуск события) в результате работы алгоритма автоматического выделения торцов бревен не превышает 5% от общего числа целевых объектов на изображении для каждого изображения.

7.1.8.4 Проверка по п. 4 Программы

Испытания программы для мобильной оценки объема штабеля круглого леса на предмет соответствия расчетных значений объема фактическим показателям, получаемым посредством ручного поштучного измерения бревен согласно ГОСТ 2708.

Проводится съемка не менее 400 различных партий леса:

- в штабелированном виде и на лесовозе;
- ориентированными к фотообъективу комлями, торцами и враскомлевку;
- в различных погодных условиях (снег, солнечная погода).

На каждом типе оборудования (Таблица 7.2) должно быть отснято не менее 50 кадров. Для 50 партий леса измерения должны быть проведены по двум изображениям (двум сторонам пакета). Для 50 партий леса выполняется серия снимков с различных расстояний.

Условия съемки: камера расположена параллельно плоскости торцов пакета бревен, пакет расположен по центру кадра и имеются отступы от крайних бревен пакета до границы кадра

Общее количество измерений – не менее 600. Выполняется последовательность действий, приведенная в Таблице 7.5.

Таблица 7.5 – Оценка корректности работы программы

№ п/п	Действие	Ожидаемый результат
1	Снимок торца или торцов штабеля леса на фотоаппарат или планшет	
2	Загрузка изображений в Программу, запуск автоматической обработки	Выделение объектов интереса с погрешностями 1го и 2го рода, не превышающими пороговых значений испытаний по п.п.1,2.
3	Ручное редактирование результатов обработки	Получение в Программе итоговых значений количества бревен и объема штабеля леса
4	Ручное измерение штабеля бревен	Получение значений количества бревен и объема штабеля леса посредством ручного поштучного измерения бревен согласно ГОСТ 2708
5	Сравнение полученных результатов	Значения объема штабеля леса отличаются не более чем на 8% от результатов ручного измерения, Значения количества бревен совпадают с результатами ручного измерения

Для случаев сравнения результатов измерения одного пакета с разных точек выполняется последовательность действий, приведенная в Таблице 7.6.

Таблица 7.6 – Оценка корректности работы программы

№ п/п	Действие	Ожидаемый результат
1	Снимок торца штабеля леса на фотоаппарат или планшет	
2	Снимок торца штабеля леса с другой точки (дистанции)	
3	Загрузка изображений в Программу, запуск автоматической обработки	Выделение объектов интереса с погрешностями 1го и 2го рода, не превышающими пороговых значений испытаний по п.п.1,2.
4	Сравнение полученных результатов	Значения объема штабеля леса отличаются не более чем на 1% для двух изображений, Значения количества

№ п/п	Действие	Ожидаемый результат
		бревен совпадают для обоих изображений измерения

Программа считается выдержавшей испытание, если после выполнения каждого действия результат соответствует приведенному в Таблицах 7.5-7.6 ожидаемому результату

7.2 Результаты испытания программы по пункту 1

Проверка работоспособности Программы в условиях промышленного предприятия в течение 554 чел/часов. В тестировании Программы было задействовано 50 пользователей.

В ходе испытаний не было выявлено критических ошибок в работе Программы. Выявленные некритические ошибки приведены в Таблице 7.7.

Таблица 7.7 – Описание некритических ошибок

№ п/п	Описание
1	Сбой при выборе обрабатываемого изображения. После подтверждения выбора изображения в процессе импорта, импортируемое изображение не загружается. В случае сбоя пользователь вынужден повторно выполнять операцию импорта изображения. Какая-либо закономерность возникновения сбоя не выявлена.
2	Проблема экспорта результата в формат PDF. При попытке экспорта результатов в формат PDF и сохранения файла результатов с именем, совпадающем с именем существующего файла, пользователь получает предложение заменить существующий файл, однако, при попытке пользователя сохранить результаты с заменой существующего файла, экспортируемый результат не сохраняется.
3	В процессе работы с приложением пропадает строка состояния. Для восстановления внешнего вида рабочего окна приложения требуется перезапуск приложения.
4	Импорт изображения разрешением 3648x2736 в формате JPG приводит к зависанию устройств обоих типов с тестируемым программным обеспечением.
5	При масштабировании изображения <60% происходит смещение границ

№ п/п	Описание
	выделенных объектов.

Общая оценка эффективности и удобства работы с Программой приведена в Таблице 7.8

Таблица 7.8 – Описание некритических ошибок

Оценка	Эффективность использования Программы для измерения объема штабеля круглого леса	Удобство пользовательского интерфейса Программы
5 (высокая)	31	28
4	14	12
3	2	7
2	1	2
1 (низкая)	2	1
средняя оценка	**4,42**	**4,28**

7.2.1 Замечания и рекомендации

В ходе опроса пользователей Программы получено 10 рекомендаций по улучшению пользовательского интерфейса и принципа работы Программы (Таблица 7.9)

Таблица 7.9 – Рекомендации

№ п/п	Описание
1	Пользователи отмечают неудобство работы с панелью инструментов, ссылаясь на размер иконок. Предлагается изменить компоновку панели инструментов, разместить однотипные инструменты управления слоями, увеличив при этом размеры иконок.
2	Предлагается реализовать в программе масштабирование с помощью сенсорного управления с всплывающей строкой для ввода точного значения масштаба.

№ п/п	Описание
3	Окно отчётов – добавить возможность визуализации информации о диаметрах бревен и их количестве не только в виде столбчатой диаграммы, а также line charts, area charts, bar charts, pie charts, scatter, bubble charts
4	Пользователи отмечают необходимость автоматизации процесса выделения эталонного бревна по маркеру (например, кресту, нанесенному мелом на торец бревна).
5	Результат работы измерения по каждому отдельному бревну нужно отображать в дополнительном окне или на информационной панели, по желанию.
6	Пользователи отмечают потребность в просмотре истории действий для возврата к определенным шагам.
7	Добавить функционал, позволяющий кадрировать изображения
8	Добавить возможность автоматического выделения группы объектов, помеченных маркером
9	Дополнить перечень доступных форматов изображений форматами PNG, PSD, HD Photo
10	Добавить возможность отправки результата по e-mail непосредственно из приложения

7.3 Результаты испытания программы по пункту 2

Проведена съемка 400 различных партий леса:

– в штабелированном виде и на лесовозе;

– ориентированными к фотообъективу комлями, торцами и враскомлевку;

– в различных погодных условиях (снег, солнечная погода).

Количество кадров для каждого типа оборудования приведено в Таблице 7.10.

Таблица 7.10 – Использование оборудования

Наименование	Кол-во кадров
Планшетный компьютер ASUS Nexus7 4 Core Snapdragon S4	236

Планшетный компьютер Samsung Galaxy Tab 3 GT0P5210 16 Gb 10,1”	396
Общее количество измерений	632

Условия съемки: камера расположена параллельно плоскости торцов пакета бревен, пакет расположен по центру кадра и имеются отступы от крайних бревен пакета до границы кадра.

Рассчитанная по итогам испытаний вероятность ошибки первого рода в результате работы алгоритма автоматического выделения торцов бревен составляет 2,74% в среднем при максимальном значении 4,56%.

Примеры полученных изображений приведены на Рисунке 7.1. Результаты работы алгоритма приведены на Рисунке 7.2 Расчеты по результатам измерений приведены в Приложении Б (Таблица Б.1).

Рисунок 7.1 – Примеры изображений

Рисунок 7.2 – Результаты работы алгоритма

7.4 Результаты испытания программы по пункту 3

Проведена съемка 400 различных партий леса:

– в штабелированном виде и на лесовозе;

– ориентированными к фотообъективу комлями, торцами и враскомлевку;

– в различных погодных условиях (снег, солнечная погода).

Количество кадров для каждого типа оборудования приведено в Таблице 7.11.

Условия съемки: камера расположена параллельно плоскости торцов пакета бревен, пакет расположен по центру кадра и имеются отступы от крайних бревен пакета до границы кадра.

Таблица 7.11 – Использование оборудования

Наименование	Кол-во кадров
Планшетный компьютер ASUS Nexus7 4 Core Snapdragon S4	246
Планшетный компьютер Samsung Galaxy Tab 3 GT0P5210 16 Gb 10,1”	358
Общее количество измерений	604

Рассчитанная по итогам испытаний вероятность ошибки второго рода в результате работы алгоритма автоматического выделения торцов бревен составляет 3,8% в среднем при максимальном значении 4,82%.

Примеры полученных изображений приведены на Рисунке 7.1. Результаты работы алгоритма приведены на Рисунке 7.2. Расчеты по результатам измерений приведены в Приложении Б (Таблица Б.1).

7.5 Результаты испытания программы по пункту 4

Проведена съемка 400 различных партий леса:

– в штабелированном виде и на лесовозе;

– ориентированными к фотообъективу комлями, торцами и враскомлевку;

– в различных погодных условиях (снег, солнечная погода).

Количество кадров для каждого типа оборудования приведено в Таблице 7.12.

Таблица 7.12 – Использование оборудования

Наименование	Кол-во кадров
Планшетный компьютер ASUS Nexus7 4 Core Snapdragon S4	350
Планшетный компьютер Samsung Galaxy Tab 3 GT0P5210 16 Gb 10,1”	590
Общее количество измерений	940

Условия съемки: камера расположена параллельно плоскости торцов пакета бревен, пакет расположен по центру кадра и имеются отступы от крайних бревен пакета до границы кадра.

Рассчитанная по итогам испытаний погрешность определения объема по сравнению с ручным методом составляет 5,14% в среднем при максимальном значении 7,09%. Погрешность в измерениях одного штабеля леса с разных точек съема составляет 1%.

Примеры полученных изображений приведены на Рисунке 7.3. Результаты работы программы приведены на Рисунке 7.4. Сравнение результатов измерений с ручным методом приведено в Приложении Б (Таблица Б.2).

Рисунок 7.3 – Примеры изображений

Рисунок7.4 – Результаты работы программы

7.6 Выводы

Испытания проводились на основании разработанной программы и методики исследовательских испытаний с использованием следующего оборудования:

– Планшетный компьютер ASUS Nexus7 4 Core Snapdragon S4

– Планшетный компьютер Samsung Galaxy Tab 3 GT0P5210 16 Gb 10,1”

В результате проведения исследовательских испытаний получены следующие результаты:

Общая продолжительность исследовательских испытаний Программы составила 554 чел/часа. За это время:

- Проведено измерений – 940 (Рисунок 7.7);
- Количество партий леса измерено – 400;
- Количество пользователей Программы – 50 чел.
- Критические ошибки в работе Программы не выявлены;
- Количество некритических ошибок – 5;
- Количество рекомендаций – 10.

В результате проведения исследовательских испытаний полученные следующие количественные характеристики Программы (Рисунки 7.5, 7.6):

- Вероятность ошибок первого рода, средняя – 2,74%;
- Вероятность ошибок первого рода, максимальная – 4,56%;
- Вероятность ошибок второго рода, средняя – 3,8%;
- Вероятность ошибок второго рода, максимальная – 4,82%;
- Отклонение значений кубатуры для разных точек съема – 1%;
- Отклонение значений кубатуры по сравнению с ручным поштучным измерением, среднее – 5,14%;
- Отклонение значений кубатуры по сравнению с ручным поштучным измерением, максимальное – 7,09% (Рисунок 7.8);
- Скорость записи в базу данных – 1,2 с
- Время загрузки страницы отчетов – 2,2 с (для 35 отчетов на страницу)
- Скорость чтения полной строки из базы данных – 9 мс
- Скорость чтения страницы из базы данных – 7 мс.
- Максимальный объем базы данных – 4398046507008 байт

Исследовательские испытания методики измерения объема штабеля круглого леса проведены в полном объеме. Работоспособность программы для мобильной оценки объема партии круглого леса подтверждена. Все качественные и количественные характеристики средства измерения,

подлежащие оценке, находятся в допустимых пределах, при этом точность автоматического расчета в большинстве случаев оказывается выше, чем у ручного метода (см .раздел 3.3).

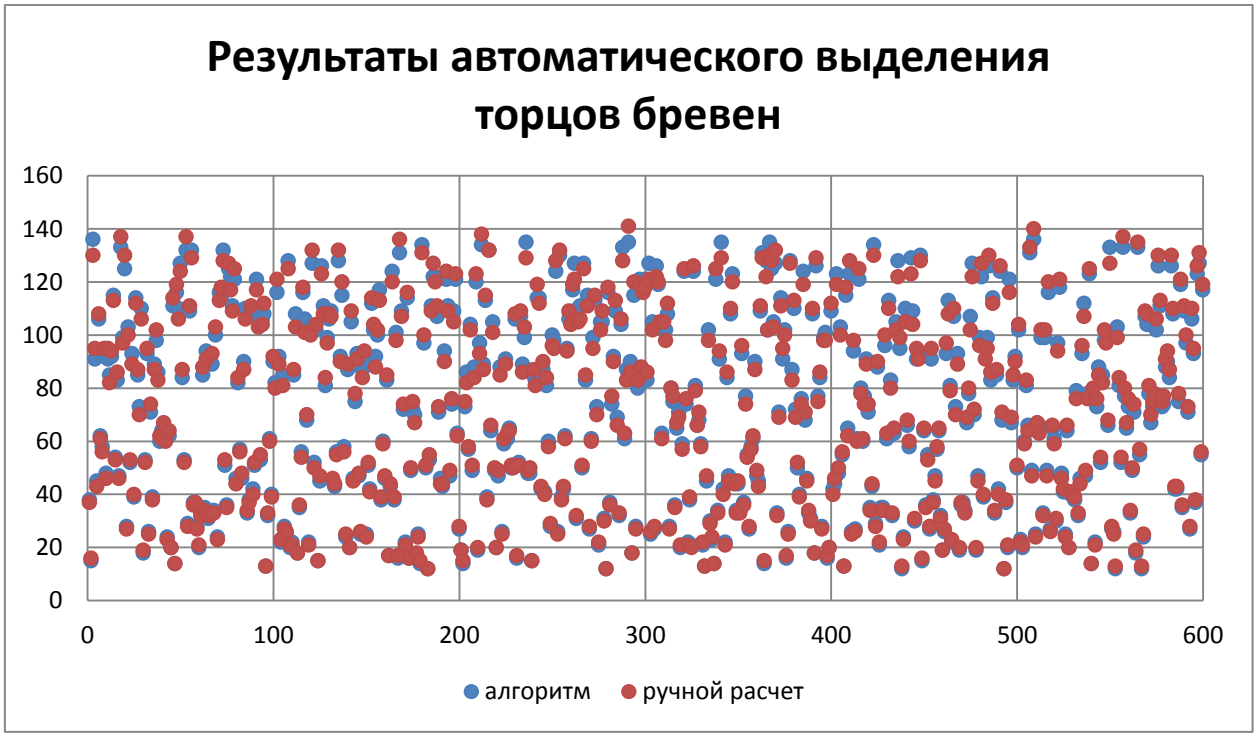


Рисунок 7.5 – Результаты автоматического выделения торцов бревен

Рисунок 7.6 – Относительная точность автоматического выделения торцов бревен

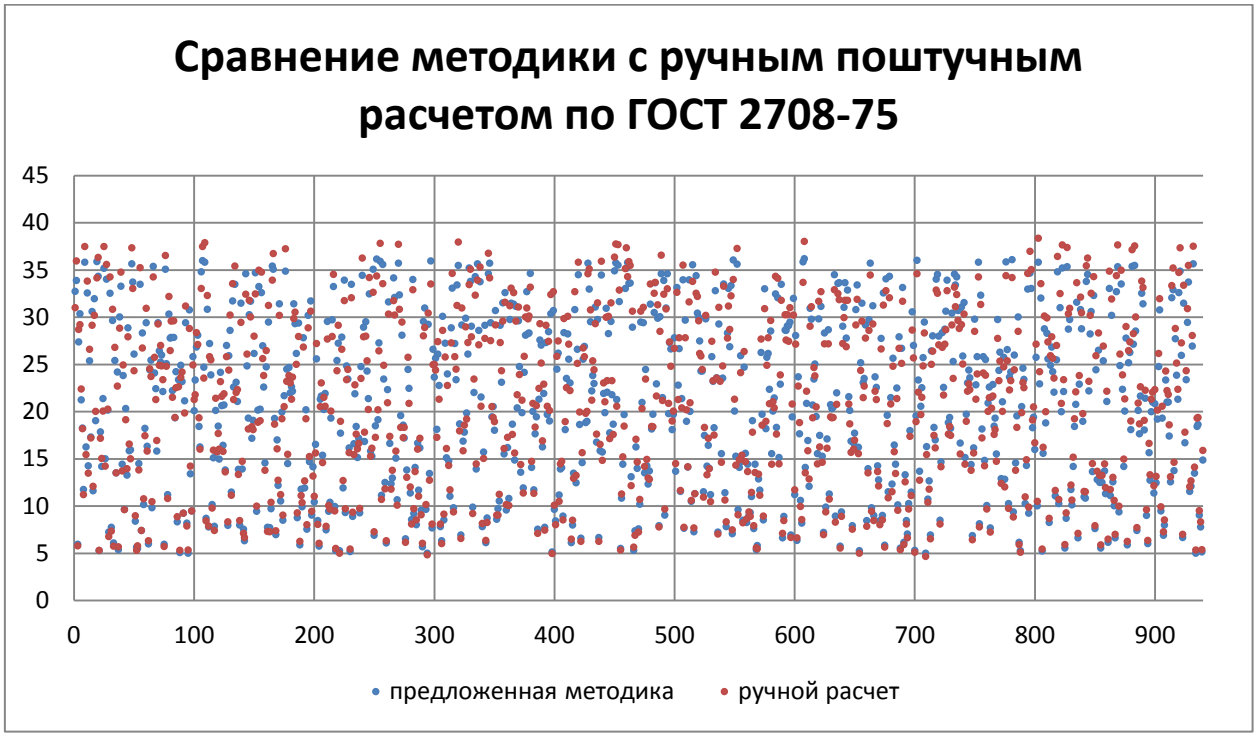


Рисунок 7.7 – Результаты сравнения методики автоматического измерения с поштучным расчетом по ГОСТ 2708-75

Рисунок 7.8 – Точность методики относительно ручного метода

ЗАКЛЮЧЕНИЕ

Анализ текущей ситуации в лесопромышленном комплексе РФ показал, что в настоящее время наибольшее распространение при погрузке/приемке и транспортировке имеют ручные способы учета объема партий круглых лесоматериалов. Для круглых лесоматериалов предельные погрешности измерения объема и контроля качества устанавливают на уровне от ±3% до ±12%, что значительно превышает предельные погрешности учета для других видов материалов. При этом методы, основанные на бесконтактном измерении не зависят от человеческого фактора и обеспечивают меньшую погрешность, однако стоимость таких решений и отсутствие мобильности делают их внедрение не всегда целесообразным.

Основные результаты диссертационной работы заключаются в следующем.

1. Разработана методика и алгоритм расчета объема партии бревен на основе фотограмметрии. Данная методика применима в лесной промышленности, и относится к области измерительной техники для определения количественных характеристик круглых лесоматериалов, может быть использована на лесозаготовительных и лесоперерабатывающих предприятиях для учета древесины а также при таможенном контроле круглых лесоматериалов. Методика заключается в фотосъемке торцов штабеля бревен цифровым устройством, получении изображений и построении модели штабеля бревен с использованием программного обеспечения. На полученном изображении, по крайней мере, одного торца штабеля производится распознавание контуров торцов всех бревен в штабеле путем отбора объектов по признакам формы и пространственного расположения и уточняются границы контуров на основе их цветовых характеристик. Далее выбирается шаблонный объект с известными

размерами для расчета калибровочных коэффициенты; с учетом которых находится площадь торцов бревен. Ориентация каждого бревна в штабеле определяется путем сопоставления полученных размеров торцов бревен. С учетом длины штабеля рассчитывается кубатура каждого бревна, путем суммирования которой определяется кубатура всего штабеля бревен.

2. Разработано программное обеспечение для мобильного расчета объема партии круглого леса. В ходе создания программы для мобильного расчета объема штабеля круглого леса решены следующие научно-технические задачи:

а. Выделение на изображении объектов, обладающих радиальной симметрией;

б. Определение принадлежности выделенного объекта к множеству торцов бревен на основе пространственных признаков;

в. Определение контура торца бревна на основе цветовых признаков;

г. Построение 3D-модели пакета бревен по изображениям торцевых срезов для определения ориентации бревен в партии и повышения точности расчетов.

д. Разработка пользовательского интерфейса, включающего:

- инструменты управления автоматической обработкой,
- инструменты визуализации 3D-модели штабеля,
- инструменты ручного редактирования результатов автоматической обработки,
- инструменты формирования отчетов.

В результате применения предложенной методики измерение объема партии круглого леса будут проводиться более оперативно, по сравнению с другими подходами (ручное измерение, взвешивание), а их результаты станут более достоверными (снижение возможности фальсификации, повторная верификация измерения). Мобильность и скорость работы данного решения позволит проводить контрольные замеры в местах заготовки леса, при погрузке на сортиментовоз, в момент отгрузки-приемки и при отправке

на переработку, то есть осуществлять учет сырья на каждом этапе его жизненного цикла. Применение разработки дает возможность автоматизации ранее недоступных для этого участков производственного процесса что приводит к повышению трудовой дисциплины персонала, снижению затрат на традиционные (бумажные) коммуникации между подразделениями компании (в среднем, на 67 %) и снижению времени операций учета (в среднем на 36 %, в диапазоне от 0 до 90 %).

Результаты диссертации были представлены на ряде научно-технических конференций по анализу и обработке изображений. Автор имеет 18 публикаций по тематике диссертации, из них 7 Scopus, 3 ВАК. На способ измерения кубатуры круглого леса получен патент № 2553714 от 22 мая 2015 г., Свидетельство о регистрации программы для ЭВМ «FoRest» № 2014611152 от 27 января 2014 г.

Программа для мобильного расчета объема штабеля круглого леса была представлена на выставках INNOPROM 2012-2014 (Екатеринбург), LESPROM-URAL Professional 2014-2015 (Екатеринбург), INTERFORST-2014 (Мюнхен, Германия), Апробация разработки проведена на предприятиях Свердловской области – ЗАО «ТУРА-ЛЕС», ООО «АРМ-Рус», ООО «ИТК-Дизель», ООО «Лазерные приборы», ООО «Энкон-сервис», ИП Козьменко, ИП Федореев. За 2016 г. общий доход от реализации результатов работы составил более 4 млн. руб. Проведение исследований и разработок на разных этапах было поддержано ИЦ Сколково, Фондом поддержки инноваций, Фондом CRDF. Разработанное программное обеспечение для мобильного расчета объема партии круглого леса отмечено премией губернатора Свердловской области в сфере информационных технологий как лучший проект 2016 года, разработанный и внедренный на предприятиях Свердловской области.

Автор выражает благодарность фонду содействия инновациям, фонду «Сколково», Свердловскому областному фонду поддержки предпринимательства, за финансовую поддержку исследований. Автор

благодарит руководство и сотрудников ЗАО «ТУРА-Лес», ЗАО «Фанком», ЗАО «Форлекс», Режевского ЛПК за помощь в исследовании и анализе производственных процессов лесопромышленных предприятий и испытаниях программы для мобильного расчета объема штабеля круглого леса.

СПИСОК ЛИТЕРАТУРЫ

1. Государственный доклад «О состоянии и об охране окружающей среды Российской Федерации в 2015 году» // URL: http://www.forestforum.ru/info/forests_2015.pdf (дата обращения 21.02.2017)

2. Лесной форум Гринпис России // URL: http://www.forestforum.ru/viewtopic.php?t=20034 (дата обращения 21.02.2017)

3. Суворов Е., Семилетова К., Овсеенко О., Щербаков., Промышленное производство по видам экономической деятельности // Сводный департамент макроэкономического прогнозирования. Департамент развития секторов экономики, декабрь 2016, URL: http://economy.gov.ru/wps/wcm/connect/5eea837f-b996-495e-a25c-9f4191cd0d76/monitor_prom12.pdf?MOD=AJPERES&CACHEID=5eea837f-b996-495e-a25c-9f4191cd0d76 (дата обращения 21.02.2017)

4. Лесная и деревообрабатывающая промышленность России // Wood-prom.ru, URL: http://wood-prom.ru/analitika/14431_lesnaya-i-derevoobrabatyvayuschaya-promyshlennost- (дата обращения 21.02.2017)

5. Карнаух М., Прибыль в инвестициях. Проблемы лесопромышленного комплекса обсуждают эксперты // Коммерсант, № 49 от 25.03.2014, URL: http://www.kommersant.ru/doc/2432307 (дата обращения 21.02.2017)

6. Оленина Т.Ю., Особенности лесного законодательства России и Финляндии // Общество и право. № 4 (46), с. 54-58, 2013 URL: http://491602.pc-forums.ru/i23.html (дата обращения 21.02.2017)

7. Внедрение информационных технологий – эффективный способ оптимизации ЛПК России // Интерактивный лесопромышленный портал FORESTEC, 28 марта 2014, URL: http://www.forestec.net/index/experts/experts_30.html (дата обращения 21.02.2017)

8. Проект Федерального закона "О государственном регулировании оборота круглых лесоматериалов и о внесении изменений в отдельные

законодательные акты РФ" // URL: http://www.mnr.gov.ru/regulatory/detail.php?ID=129090 (дата обращения 21.02.2017)

9. Пособие по учету круглых лесоматериалов // Лесэксперт, Проект 2012-08-05, URL: http://www.lesexpert.info/2012-10-15-roundwood_handbook-33.pdf (дата обращения 21.02.2017)

10. Тюрин Н.А., Бессараб Г.А., Кочанов В.В., Использование электронных тахеометров при определении объемов штабелей круглых лесоматериалов // ЛесПромИнформ №4 (35), 2006, URL: http://www.lesprominform.ru/jarchive/articles/itemshow/1450 (дата обращения: 21.02.2017)

11. The Benefits of MES: A Report from the Field // MESA International, URL: http://www.cpdee.ufmg.br/~seixas/PaginaII/Download/DownloadFiles/pap1.pdf (дата обращения: 21.02.2017)

12. Самойлов А.Н., Классификация и определение основных направлений развития методов измерения объема круглого лесоматериала // Научный журнал КубГАУ, № 24(8), 2006

13. Бит Ю.А., Вавилов С.В., Измерения объемов круглого леса. Справочник // СПб.: ПРОФИКС, 2008, 360 с.

14. Петровский В.С., Автоматизация лесопромышленных предприятий // М.: Издательский центр «Академия», 2005, 304 с.

15. Николаев Д.П., Котов А.А., Усилин С.А., Построение устойчивых признаков для алгоритма Виолы и Джонса в задаче классификации транспортных средств // Труды 35-ой конференции молодых ученых и специалистов «Информационные технологии и системы (ИТиС’12)». Петрозаводск, 2012. М.: ИППИ РАН, 2012., с. 383-388

16. Буй Тхи Тху Чанг, Фан Нгок Хоанг, Спицын В. Г., Распознавание лиц на основе применения метода Виолы-Джонса, вейвлет-преобразования и метода главных компонент // Известия Томского политехнического

университета [Известия ТПУ], 2012. Т. 320, № 5 : Управление, вычислительная техника и информатика. С. 54-59.

17. Андреева А.А., Иванов А.Л., Обнаружение лица на изображении в автоматизированных системах габитоскопической идентификации // Вестник чувашского университета, 2006, Чебоксары, № 2 – с. 316-322

18. Муравьев В.С., Муравьев С.И. Адаптивный алгоритм выделения и обнаружения воздушных объектов на изображениях // Информационно-управляющие системы, 2011. СПб., № 5, с. 8-14

19. Чочиа П. А. Пирамидальный алгоритм сегментации изображений // Информацион- ные процессы. 2010. Т. 10, № 1. С. 23–35

20. Dasgupta S., Das S., Biswas A., Abraham A., Automatic circle detection on digital images with an adaptive bacterial foraging algorithm // Soft. Computing 14 (11), 2009, pp. 1151-1164

21. Мишустин С.М., Исследование алгоритма BFOA в рамках задачи распознавания простых фигур на изображении // Материалы IV международной научно-технической конференции студентов, аспирантов и молодых ученых. – Донецк, ДонНТУ – 2013

22. Садыков С.С., Терехин А.В., Алгоритм классификации выпуклых фигур с использованием диагональных признаков формы // Вестник КГУ им. Н.А. Некрасова. – 2013. – №6. – С. 13-17.

23. Минкина А., Григорьев А., Усилин С., Полевой Д., Николаев Д.. Обобщение метода Виолы и Джонса в виде решающего дерева сильных классификаторов для распознавания объектов в видеопотоке в режиме реального времени // Информационные технологии и системы. Сборник трудов. 2014. С. 303-307.

24. Turker M., Koc-San D., Building extraction from high-resolution optical spaceborne images using the integration of support vector machine (SVM) classification, Hough transformation and perceptual grouping. // Int. J. Applied Earth Observation and Geoinformation Vol. 34, pp. 58-69, 2015

25. Котов А., Николаев Д., Прослеживание в видеопотоке объектов, содержащих множество концентрических дуг URL: http://itas2012.iitp.ru/pdf/1569605095.pdf (lдата обращения 21.02.2017)

26. Hahn K-S., Jung S., Han Y., Hahn H., A new algorithm for ellipse detection by curve segments // Pattern Recognition Letters vol. 29(13): pp. 1836-1841, 2008

27. F. Mai, Y. S. Hung, H. Zhong, W. F. Sze, A hierarchical approach for fast and robust ellipse extraction, Pattern Recognition, vol. 41, no. 8, pp. 2512–2524, 2008

28. De Marco T., Cazzato, D., Marco L., Cosimo D., Randomized circle detection with isophotes curvature analysis // Pattern Recognition, vol. 48 (2), pp.411-421, 2015

29. Rad A.A., Faez K., Qaragozlou N., Fast circle detection using gradient pair vectors // In Proceedings of the Seventh International Conference on DICTA, pp. 10-12, 2003

30. Yin P., Chen L., New method for ellipse detection by means for symmetry // Journal of Electronic Imaging. vol. 3(1), pp. 20-29, 1994

31. Breiman L. Random forests // Machine Learning, 45(1): 5–32, 2001.

32. Dalal N., Triggs B. Histograms of oriented gradients for human detection // IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), San Diego, CA, USA, 2005, pp. 886-893 vol. 1.

33. Déniz O., Bueno G., Salido J., De la Torre F. Face recognition using Histograms of Oriented Gradients // Pattern Recogn. Lett. 32, 12 (September 2011), 1598-1603.

34. Galsgaard B., Lundtoft D.H., Nikolov I., Nasrollahi K., Moeslund T.B. Circular Hough Transform and Local Circularity Measure for Weight Estimation of a Graph-Cut Based Wood Stack Measurement // IEEE Winter Conference on Applications of Computer Vision, Waikoloa, HI, 2015, pp. 686-693.

35. Gutzeit E., Voskamp J. Automatic segmentation of wood logs by combining detection and segmentation // International Symposium on Visual Computing. pp. 252–261 (2012).

36. Herbon C., Tönnies K., Stock B. Detection and segmentation of clustered objects by using iterative classification, segmentation, and Gaussian mixture models and application to wood log detection // Pattern Recognition. Springer International Publishing, (2014), pp. 354-364.

37. Herbon C. The HAWKwood Database // CoRR abs/1410.4393 (2014): n. pag.

38. Knyaz V.A., Maksimov A.A. Photogrammetric Technique for Timber Stack Volume Contol // International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Vol. XL-3, 157-162, 2014.

39. Knyaz V.A., Vizilter Yu.V., Zheltov S.Yu. Photogrammetric techniques for measurements in woodworking industry // International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences Proceedings, Vol. XXXIII, part B5/2, pp. 42-47, 2004.

40. Misra A., Takashi A., Okatani T., Deguchi K. Hand Gesture Recognition Using Histogram of Oriented Gradients and Partial Least Squares Regression // MVA2011, IAPR Conference on Machine Vision Applications; 2011; Nara, Japan.

41. Mullin M., Sukthankar R. Complete Cross-Validation for Nearest Neighbor Classifiers // Proceedings of International Conference on Machine Learning. 2000. C. 1137-1145.

42. Ojala T., Pietikainen M., Harwood D. Performance evaluation of texture measures with classification based on Kullback discrimination of distributions // Proceedings of the 12th IAPR International Conference on Pattern Recognition, 1994, vol. 1 - Conference A: Computer Vision & Image Processing, vol. 1, pp. 582–585, Oct 1994.

43. Viola P., Jones M. Rapid object detection using a boosted cascade of simple features // Proc. Int. Conf. on Computer Vision and Pattern Recognition. 2001. № 1. P. 511−518.

44. Wang X., Doretto G., Sebastian T.B., Rittscher J., Tu P.H. Shape and Appearance Context Modeling // Proceedings of IEEE International Conference on Computer Vision (ICCV) 2007.

45. Cheng H.D., Guo Y., Zhang Y., A novel Hough transform based on eliminating particle swarm optimization and its applications // Pattern Recognition, vol. 42 (9), pp. 1959-1969, 2009

46. Cauchie J., Fiolet V., Villers D., Optimization of an Hough transform algorithm for the search of a center // Pattern Recognition vol. 41(2): pp. 567-574, 2008

47. Mochizuki Y., Torii A., Imiya A., N-Point Hough transform for line detection. // J. Visual Communication and Image Representation, vol. 20(4): pp. 242-253, 2009

48. Fornaciari M., Prati A., Cucchiara R., A fast and effective ellipse detector for embedded vision applications // Pattern Recognition vol. 47(11) pp. 3693–3708, 2014

49. Pan L., Chu W.S., Saragih J.M., De la Torre F., Xie M., Fast and Robust Circular Object Detection with Probabilistic Pairwise Voting (PPV) // IEEE Signal Processing Letters, vol. 18, no. 11, pp. 639-642, 2011

50. Šukilović T., Curvature based shape detection // Computational Geometry, vol. 48 (3), pp. 180-188, 2015

51. Ioannou D., Huda W., Laine A.F., Circle recognition through a 2D Hough Transform and radius histogramming // Image and Vision Computing 17 (1), pp. 15-26, 1999

52. Hassanein A.S., Mohammad S., Sameer M., Ragab M.E., A Survey on Hough Transform, Theory, Techniques and Applications. CoRR abs, pp. 1-18, 2015

53. Prasad D.K., Leung M.K.H., Cho S.Y., Edge curvature and convexity based ellipse detection method // Pattern Recognition, vol. 45(9):pp. 3204-3221, 2012

54. Лебедев С.А., Ососков Г.А., Быстрые алгоритмы распознавания колец и идентификации электронов в детекторе RICH эксперимента CBM // Письма в ЭЧАЯ. 2009. Т. 6, № 2(151). С. 260-284.

55. Marin D., Gegundez-Arias M.E., Suero A., Bravo J.M., Obtaining optic disc center and pixel region by automatic thresholding methods on morphologically

processed fundus images // Proceedings Of Computer Methods And Programs In Biomedicine, Elsevier, Vol.118, No.2, Page No.173-185, 2014.

56. Lu W., Tan J., Detection of incomplete ellipse in images with strong noise by iterative randomized hough transform (IRHT) // Pattern Recognition, vol. 41, no. 4, pp. 1268 – 1279, 2008

57. Kierkegaard P., A method for detection of circular arcs based on the Hough transform // Machine Vision and Applications, vol. 5 (4), pp. 249-263, 1992

58. Марр Д. Зрение: Информационный подход к изучению представления и обработки зрительных образов. М., «Радио и связь», 1987

59. Loy, G., Zelinsky, A., Fast radial symmetry for detecting points of interest // IEEE Transactions on PAMI, vol. 25, no. 8, 2003, pp. 959-973

60. Damien Rj., The mean shift clustering algorithm // EFAVDB, URL: http://efavdb.com/mean-shift/#Tech (дата обращения 21.02.2017)

61. Comaniciu D., Meer P., Mean Shift: A Robust Approach Toward Feature Space Analysis // In IEEE Transactions on PAMI, vol. 24 No 5, pp. 603-619, 2002

62. Мандель И.Д., Кластерный анализ. М.: Финансы и Статистика, 1988, 186 с.

63. Скворцов А.В. Триангуляция Делоне и её применение // Томск: Изд-во Том. ун-та, 2002. – 128 с.

64. Скворцов А.В., Мирза Н.С., Алгоритмы построения и анализа триангуляции. – Томск: Изд- во Том. ун-та, 2006. – 168 с.

65. Ерзин А. И., Кочетов Ю. А., Задачи маршрутизации : учеб. пособие / Новосиб. гос. ун-т. – Новосибирск : РИЦ НГУ, 2014. – 95 с.

66. Шапиро Л., Стокман Дж., Компьютерное зрение // М.: БИНОМ. Лаборатория знаний 2006, 752 с.

67. Гонсалес Р., Вудс Р., Цифровая обработка изображений // М: Техносфера, 2005, 1072 с.

68. Форсайт Д., Понс Ж., Компьютерное зрение. Современный подход // М.: Издательский дом «Вильямс», 2004, 928 с.

69. Kass M., Witkin A., Terzopoulos D., Snakes: Active Contour Models // International Journal of Computer Vision, pp. 321-331, 1987

70. Laurent D.C., On Active Contour Models and Balloons // In CVGIP:IU, vol. 53(2), pp. 1-18, 1991

71. Xu C., Prince J.L., Snakes, Shapes, and Gradient Vector Flow // IEEE Trans. Image Processing, vol. 7, no. 3, pp. 359-369, 1998

72. Velasco F.A., Marroquin J.L., Robust parametric active contours: the Sandwich Snakes // Machine Vision and Applications, vol. 12, pp.238-242, 2001

73. Chen H., Introduction to Min-Cut/Max-Flow Algorithms // UCLA CIVS, URL: http://perso.telecom-paristech.fr/~tupin/cours/matim/articles/theorie_gra.pdf (дата обращения 21.02.2017)

74. Алгоритмы минимизации энергии на основе разрезов графов // URL: http://www.machinelearning.ru/wiki/images/9/9e/GM_graphCuts.pdf (дата обращения: 21.02.2017)

75. Boykov Y., Kolmogorov V., An Experimental Comparison of Min-Cut/Max-Flow Algorithms for Energy Minimization in Vision // In IEEE Transactions on PAMI, vol. 26, No. 9, pp. 1124-1137, 2004

76. Sudipta N. S., Graph Cut Algorithms in Vision, Graphics and Machine Learning, 2004

77. Roerdink J.B.T.M., Meijster A., The Watershed Transform: Definitions, Algorithms and Parallelization Strategies // Fundamenta Informaticae, vol. 41, IOS Press, pp. 187–228, 2001

78. Г. Вагнер, Основы исследования операций Том 1. Пер. с англ.- Издательство «Мир», М., 1972

79. ГОСТ 32594-2013 Лесоматериалы круглые. Методы измерений

80. ГОСТ 2708-75 Лесоматериалы круглые. Таблицы объемов

81. Круглов А.В., Чирышев Ю.В., Детектор движущихся объектов в задаче оценки параметров круглого леса // Фундаментальные исследования, № 11, 2013, с. 915-918

82. Круглов А.В., Круглов В.Н., Чирышев Ю.В., Чирышев А.В., Применение ресурсоемких алгоритмов в системах машинного зрения реального времени // Фундаментальные исследования, № 10, 2013, с. 2612-2619.

83. Круглов А.В., Доросинский Л.Г., Чирышев Ю.В., Determination of geometric properties of the solid body from a single view in case of the timber cubature estimation // 4th International Academic Conference on Applied and Fundamental Studies, 2013, с. 255- 259

84. Круглов А.В., Югфельд И.Д., Проектирование базы данных для приложения расчета кубатуры круглого леса под управлением платформы Android // Программные системы и вычислительные методы. – 2014. – № 4. – с.431-436.

85. Круглов А.В., Круглов В.Н., Чирышев Ю.В., Determination of Direction and Velocity of the Objects // Proceedings of IMTA-5 2015 5th International Workshop on Image Mining. Theory an Applications In conjunction with 10th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - VISIGRAPP -2015, Berlin, Germany, 2015, с. 103-107

86. Сысолетин Е.Г., Аксенов К.А., Круглов А.В., Интеграция гетерогенных информационных систем современного промышленного предприятия // Современные проблемы науки и образования. – 2015. – № 1

87. Круглов А.В., Гизаттулина А.Р., Тюленева К. В., Югфельд И.Д., Сравнительный анализ алгоритмов выделения на цифровых изображениях объектов эллипсоидной формы // International research journal №5 (36) 2015, с. 81-83

88. Круглов А.В., Чирышев Ю.В., Comparative Analysis of Methods for the Log Boundaries Isolation // 12th International Conference on Informatics in Control, Automation and Robotics (ICINCO 2015), Colmar, France, 2015, с. 357-361

89. Круглов А.В., Югфельд И.Д., Реализация интерактивной сегментации для сенсорных устройств на базе ОС Android // Современные наукоемкие технологии № 2 (2) 2016, с. 229-234

90. Круглов А.В., Development of the rounded objects automatic detection method for the log deck volume measurement // Proc. SPIE 10011, First International Workshop on Pattern Recognition, 1001104 (July 11, 2016); doi:10.1117/12.2242172

91. Круглов А.В., Чирышев Ю.В., Detection and Tracking of the Objects in Real Time Applied to the Problem of the Log Volume Estimation // Supplementary Proceedings of the Fifth International Conference on Analysis of Images, Social Networks and Texts (AIST 2016), Yekaterinburg, Russia, pp. 114-123, 2016

92. Круглов А.В., Чирышев А.В., Шишко Е.В., Applying of the NVIDIA CUDA to the video processing in the task of the roundwood volume estimation // ITM Web Conf. Volume 8, 2016

93. Круглов А.В., The Algorithm of the Roundwood Volume Measurement via Photogrammetry // Proceedings of 2016 International Conference DICTA; Gold Coast; Australia, с. 399-403

94. Круглов А.В., Югфельд И.Д. The algorithm for automatic detection of the calibration object. AIP Conference Proceedings vol. 1836, issue 1; doi: http://dx.doi.org/10.1063/1.4982010, 2017

95. Круглов А.В., Чирышев Ю.В. The Image Analysis Algorithm for the Log Pile Photogrammetry Measurement. WSEAS Transactions on Signal Processing, E-ISSN: 1790-5052 / 2224-3488, Volume 13, 2017, Art. #15, pp. 135-145

96. Круглов А.В., Шишко Е.В., Кожова В.А., Завада С.Г. New Method and Software for the Round Timber Automatic Measurement. NAUN International journal of energy and environment, vol. 11, 2017 pp. 42-49

ПРИЛОЖЕНИЕ А

Обзор различных методов измерений древесины, используемых сегодня в целлюлозно-бумажной промышленности в мире – Христиан Пассот, Сантьяго, Чили.

Христиан представил различные методики измерения балансов по всему миру. Он охарактеризовал в общих чертах методы измерения балансов, главным образом в штабелях, но также и измерения по весу. Он показал, что штабельной мерой можно манипулировать при укладке бревен в штабель. Что бревна при хранении в течение 6 недель могут потерять до 25 % веса (в зависимости от погодных условий и порода). Преимущества и недостатки каждого метода были обсуждены, и они включали примеры древесины, включая применение сухой массы и измерение веса при погружении в воду.

Перевод презентации подготовлен ООО «Лесэксперт». Представлены выдержки из презентации, оформление на русском языке выполнено ООО «Квинта»

Почему измерение леса является важным?

Стоимость древесины:

- → > 50% Стоимости продукции
- → $20-100 млн./год (Чили)

Примерное распределение стоимости:

- 1/3 стоимости = древесина
- 1/3 стоимости = заготовка
- 1/3 стоимости = транспортирование

Важно правильно измерять древесину

даже если лес принадлежит заводу

Основные методы и единицы измерения

Традиционные методы измерения древесины...

Основные методы и единицы измерения

- Штабельный метод

Основные методы и единицы измерения

- Измерение по весу

Основные методы и единицы измерения

- Поштучное измерение

WOODTECH

Некоторые последствия: проблемы отрасли

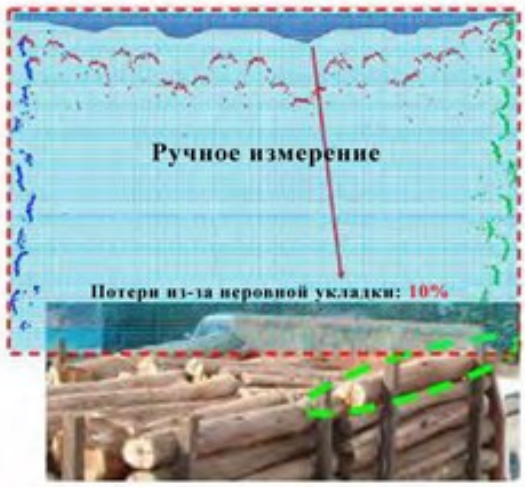
- Стоимость измерений древесины
- Важны различия между:
  - Объем заготовок ←→ Древесина на заводе
  - Поставленной древесиной ←→ Оплаченной древесиной/запасами на заводе
- Порочные стимулы → Несправдливая оплата
- Риски обмана

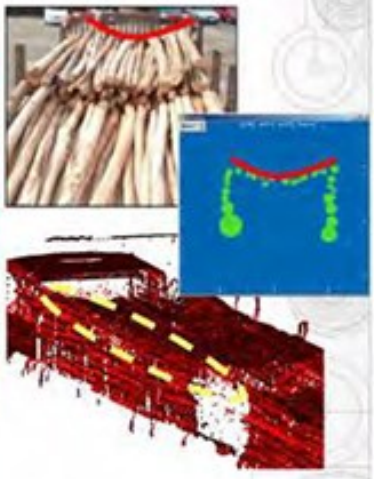
Существенное влияние на стоимость древесины → на стоимость производства

Примеры обмана

Существенное влияние на стоимость древесины → на стоимость производства

Примеры обмана





Какие меры предпринять?

"Показатель количества должен быть ОБЪЕКТИВНЫМ, ВОСПРОИЗВОДИМЫМ, ЛЕГКО И ЭФФЕКТИВНО ПО СТОИМОСТИ ОПРЕДЕЛЯЕМЫМ и СПРАВДЕЛИВЫМ и для покупателя и для продавца"

Измерение количества балансов, Russell Morkel (1998)

Единица · Метод

1. **Объективность:** не зависит от человеческих факторов
2. **Воспроизводимость:** нет изменений под внешними факторами
3. **Эффективность:** быстрые измерения и с низкой стоимостью
4. **Справдливость:** равноправные измерения для обеих сторон
5. **Стимул:** не создают порочных стимулов (для обмана)

D-401

Сравнение единиц

Мера должна быть непосредственно связана с ценностью продаваемой продукции

Сырая древесина = Сухая древесина + Биометрические характеристики + Вода

нет ценности

ценность

Цель в том, чтобы измерять только ценность

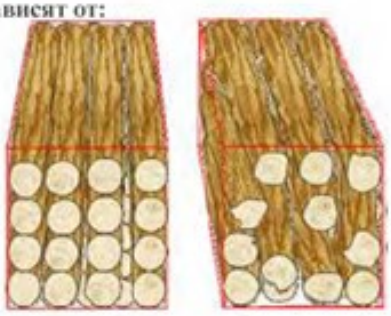
D-401-01


Оценка древесины по весу
Плотность древесины влияет на Влажность
Kg/m³
Вес воды
Общий сырой вес
Вес волокна
Низкая
Высокая
Плотность
60%
40%
30%
70%
Низкая плотность +
Высокая влажность
Высокаялотность +
Низкая влажность
ВЕС
≈
ВЕС
Ценность для процесса ≠ Ценность для процесса


Сравнение единиц - ВЕС
→ Вес не связан с ЦЕННОСТЬЮ - вода и плотность
→ Измерение веса не воспоспроизводится по времени - сушка
→ Вес производит порочные стимулы - транспортные затраты
ЛЕСОЗАВОДЫ:
→ Вес не предоставляет информацию о биометрии древесины
ТОТ ЖЕ ВЕС
РАЗЛИЧНАЯ ЦЕННОСТЬ
ВЕС не адекватная единица для измерения древесины

Оценка древесины по рамочному объему

- Измерение рамочного объема включает пространства между бревнами
- Воздушные пространства зависят от:
  - Распределении диаметра
  - Расположении бревен в штабеле
  - Длины регистраций
  - Сучков, кривизны



РАМОЧНЫЙ ОБЪЕМ не адекватная единица для измерения древесины

D-401-01

Сравнение единиц

Плотный объем

ПЛОТНЫЙ ОБЪЕМ

ВОЛОКНО
ЦЕННОСТЬ

ВОДА
НЕТ ЦЕННОСТИ

... НО ОБЪЕМ ОСТАЕТСЯ ПОСТОЯННЫМ

*"Самое большое преимущество измерения балансов по **объему** состоит в том, что объем остается **неизменным** [...] от пункта измерений до пункта переработки"*

Измерения количества балансов, Russell Morkel (1998)

ПЛОТНЫЙ ОБЪЕМ - разумная единица для измерения древесины ....., но эта единица не легкая для измерения

D-401-01

Косвенные методы измерения
Погружение

- НЕТОЧНОСТЬ ИЗМЕРЕНИЯ
  - Изменение погружения захвата
  - Динамические силы в воде и вне воды
  - Вода с плавающим материалом

- Сложная операция
  - Медленная
  - Высокие эксплуатационные затраты
  - Процесс непроверяемый

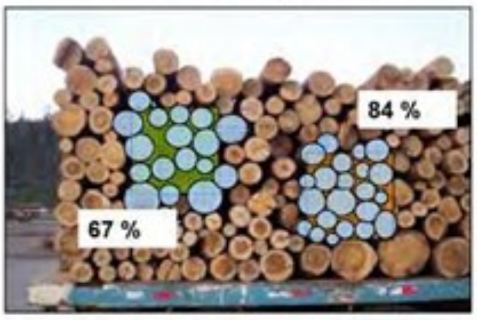
Движение захвата в воду и из воды создает динамические усилия, которые искажают измерение

Плавающие части древесины изменяют измеряемый объем

Погружение захвата должно всегда оставаться постоянным, или иначе это влияет на измеряемый объем

Косвенные методы измерения
От рамного к плотному

Анализ изображения



Конверсионный фактор зависит **от выбранного участка** для анализа изображения, который является **представительной** выборкой.

Сравнение методов

Рамочный объем + Измерение штабеля

- Используют в некоторых скандинавских странах
- Коэффициенты вариации коэффициентов полнодревесности и коэффициентов плотности также сопоставимы – (3÷7)%.
- Погрешности измерения штабельного (рамочного) объема и массы партии незначительные и сопоставимые - около ±3 %.
- Больше требований к ресурсам, чем у весовой выборочной системы, но при этом достигаются более достоверные результаты

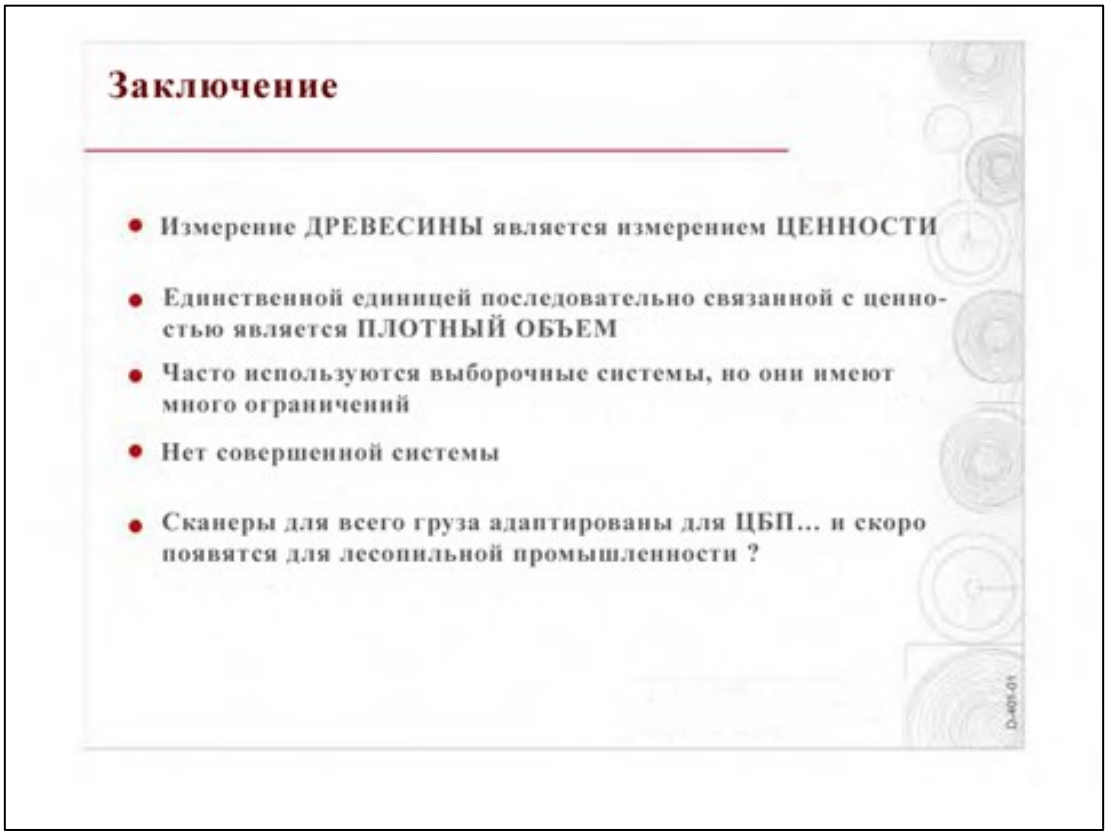
D-401-01

Сравнение методов

Измерение веса + поштучное измерение

- Вес имеет плохую связь с объемом
- Риск проблем всякий раз, когда порода изменяется (Горная сосна, и т.д)
- Легко манипулировать, порочные стимулы
- Поштучные измерения дорогие

D-401-01


Заключение
Измерение ДРЕВЕСИНЫ является измерением ЦЕННОСТИ
Единственной единицей последовательно связанной с ценностью является ПЛОТНЫЙ ОБЪЕМ
Часто используются выборочные системы, но они имеют много ограничений
Нет совершенной системы
Сканеры для всего груза адаптированы для ЦБП... и скоро появятся для лесопильной промышленности ?

ПРИЛОЖЕНИЕ Б

Б.1 Результаты проверки качества автоматического выделения торцов бревен на изображении

Таблица Б.1 – Результаты испытаний

№ кадра	алгоритм	фактическое значение	относительная точность	№ кадра	алгоритм	фактическое значение	относительная точность
1	38	37	0,973684	**301**	83	86	1,036145
2	15	16	1,066667	**302**	127	121	0,952756
3	136	130	0,955882	**303**	25	26	1,04
4	91	95	1,043956	**304**	105	102	0,971429
5	45	43	0,955556	**305**	27	28	1,037037
6	106	108	1,018868	**306**	126	122	0,968254
7	62	61	0,983871	**307**	119	120	1,008403
8	58	56	0,965517	**308**	103	105	1,019417
9	92	95	1,032609	**309**	63	61	0,968254
10	48	46	0,958333	**310**	102	105	1,029412
11	91	95	1,043956	**311**	102	98	0,960784
12	85	82	0,964706	**312**	108	112	1,037037
13	92	94	1,021739	**313**	28	27	0,964286
14	115	113	0,982609	**314**	80	80	1
15	54	53	0,981481	**315**	75	77	1,026667
16	83	86	1,036145	**316**	36	35	0,972222
17	47	46	0,978723	**317**	65	67	1,030769
18	133	137	1,030075	**318**	71	69	0,971831
19	99	97	0,979798	**319**	20	21	1,05
20	125	130	1,04	**320**	59	57	0,966102
21	28	27	0,964286	**321**	124	125	1,008065
22	103	100	0,970874	**322**	74	76	1,027027
23	52	53	1,019231	**323**	22	21	0,954545
24	93	89	0,956989	**324**	38	39	1,026316
25	39	40	1,025641	**325**	21	20	0,952381
26	114	112	0,982456	**326**	124	126	1,016129
27	85	87	1,023529	**327**	81	79	0,975309
28	73	70	0,958904	**328**	66	66	1
29	110	106	0,963636	**329**	68	71	1,044118
30	18	19	1,055556	**330**	59	58	0,983051
31	53	52	0,981132	**331**	21	22	1,047619
32	93	95	1,021505	**332**	13	13	1
33	26	25	0,961538	**333**	45	47	1,044444
34	71	74	1,042254	**334**	102	98	0,960784

№ кадра	алгоритм	фактическое значение	относительная точность	№ кадра	алгоритм	фактическое значение	относительная точность
35	39	38	0,974359	**335**	30	29	0,966667
36	89	87	0,977528	**336**	23	24	1,043478
37	98	102	1,040816	**337**	14	14	1
38	86	83	0,965116	**338**	121	125	1,033058
39	60	62	1,033333	**339**	34	33	0,970588
40	64	61	0,953125	**340**	91	94	1,032967
41	65	67	1,030769	**341**	135	129	0,955556
42	62	60	0,967742	**342**	42	40	0,952381
43	24	23	0,958333	**343**	22	21	0,954545
44	62	64	1,032258	**344**	84	86	1,02381
45	20	20	1	**345**	47	45	0,957447
46	111	114	1,027027	**346**	108	110	1,018519
47	14	14	1	**347**	123	120	0,97561
48	116	119	1,025862	**348**	45	44	0,977778
49	110	106	0,963636	**349**	34	33	0,970588
50	127	124	0,976378	**350**	44	45	1,022727
51	84	87	1,035714	**351**	34	33	0,970588
52	53	52	0,981132	**352**	93	96	1,032258
53	132	137	1,037879	**353**	37	36	0,972973
54	29	28	0,965517	**354**	77	74	0,961039
55	109	111	1,018349	**355**	55	54	0,981818
56	132	129	0,977273	**356**	27	28	1,037037
57	37	36	0,972973	**357**	58	57	0,982759
58	36	37	1,027778	**358**	61	62	1,016393
59	28	27	0,964286	**359**	90	87	0,966667
60	20	21	1,05	**360**	47	49	1,042553
61	33	32	0,969697	**361**	45	43	0,955556
62	85	88	1,035294	**362**	109	111	1,018349
63	35	34	0,971429	**363**	131	129	0,984733
64	94	91	0,968085	**364**	14	15	1,071429
65	31	32	1,032258	**365**	128	122	0,953125
66	33	32	0,969697	**366**	102	102	1
67	89	93	1,044944	**367**	135	130	0,962963
68	34	33	0,970588	**368**	125	128	1,024
69	100	103	1,03	**369**	105	103	0,980952
70	24	23	0,958333	**370**	127	132	1,03937
71	116	113	0,974138	**371**	33	32	0,969697
72	114	118	1,035088	**372**	71	69	0,971831
73	132	128	0,969697	**373**	114	111	0,973684
74	51	53	1,039216	**374**	91	95	1,043956
75	36	35	0,972222	**375**	102	100	0,980392
76	125	127	1,016	**376**	16	17	1,0625

№ кадра	алгоритм	фактическое значение	относительная точность	№ кадра	алгоритм	фактическое значение	относительная точность
77	122	117	0,959016	**377**	26	25	0,961538
78	111	109	0,981982	**378**	128	127	0,992188
79	121	125	1,033058	**379**	87	83	0,954023
80	46	44	0,956522	**380**	110	113	1,027273
81	82	84	1,02439	**381**	72	69	0,958333
82	57	56	0,982456	**382**	50	52	1,04
83	46	48	1,043478	**383**	40	39	0,975
84	90	87	0,966667	**384**	76	74	0,973684
85	110	106	0,963636	**385**	124	119	0,959677
86	33	34	1,030303	**386**	68	71	1,044118
87	38	37	0,973684	**387**	46	45	0,978261
88	107	111	1,037383	**388**	33	34	1,030303
89	42	40	0,952381	**389**	31	30	0,967742
90	51	52	1,019608	**390**	108	110	1,018519
91	121	117	0,966942	**391**	18	18	1
92	106	103	0,971698	**392**	126	129	1,02381
93	53	55	1,037736	**393**	77	75	0,974026
94	106	104	0,981132	**394**	84	86	1,02381
95	108	112	1,037037	**395**	28	27	0,964286
96	13	13	1	**396**	99	98	0,989899
97	32	33	1,03125	**397**	101	98	0,970297
98	61	60	0,983607	**398**	16	17	1,0625
99	40	39	0,975	**399**	20	20	1
100	90	92	1,022222	**400**	109	112	1,027523
101	82	80	0,97561	**401**	42	40	0,952381
102	116	121	1,043103	**402**	44	46	1,045455
103	92	89	0,967391	**403**	123	119	0,96748
104	22	23	1,045455	**404**	48	50	1,041667
105	85	81	0,952941	**405**	103	100	0,970874
106	28	27	0,964286	**406**	55	56	1,018182
107	25	26	1,04	**407**	13	13	1
108	128	125	0,976563	**408**	115	118	1,026087
109	20	21	1,05	**409**	65	62	0,953846
110	22	21	0,954545	**410**	123	128	1,04065
111	85	87	1,023529	**411**	26	25	0,961538
112	108	103	0,953704	**412**	94	98	1,042553
113	18	18	1	**413**	27	26	0,962963
114	35	36	1,028571	**414**	61	60	0,983607
115	56	54	0,964286	**415**	121	125	1,033058
116	116	118	1,017241	**416**	80	78	0,975
117	106	101	0,95283	**417**	60	61	1,016667
118	68	70	1,029412	**418**	77	74	0,961039

№ кадра	алгоритм	фактическое значение	относительная точность	№ кадра	алгоритм	фактическое значение	относительная точность
119	22	21	0,954545	**419**	91	89	0,978022
120	103	100	0,970874	**420**	71	74	1,042254
121	127	132	1,03937	**421**	35	34	0,971429
122	52	50	0,961538	**422**	43	44	1,023256
123	102	104	1,019608	**423**	134	130	0,970149
124	15	15	1	**424**	29	28	0,965517
125	45	47	1,044444	**425**	88	90	1,022727
126	126	123	0,97619	**426**	21	22	1,047619
127	111	108	0,972973	**427**	35	34	0,971429
128	81	84	1,037037	**428**	35	34	0,971429
129	99	97	0,979798	**429**	96	100	1,041667
130	106	108	1,018868	**430**	61	63	1,032787
131	109	107	0,981651	**431**	113	110	0,973451
132	45	46	1,022222	**432**	83	80	0,963855
133	43	44	1,023256	**433**	32	33	1,03125
134	56	55	0,982143	**434**	63	65	1,031746
135	128	132	1,03125	**435**	105	102	0,971429
136	92	90	0,978261	**436**	128	122	0,953125
137	115	120	1,043478	**437**	95	99	1,042105
138	58	56	0,965517	**438**	12	13	1,083333
139	24	25	1,041667	**439**	24	23	0,958333
140	87	89	1,022989	**440**	110	105	0,954545
141	20	20	1	**441**	66	68	1,030303
142	105	109	1,038095	**442**	58	60	1,034483
143	46	45	0,978261	**443**	129	123	0,953488
144	75	78	1,04	**444**	109	104	0,954128
145	93	91	0,978495	**445**	30	31	1,033333
146	47	48	1,021277	**446**	91	95	1,043956
147	25	26	1,04	**447**	93	91	0,978495
148	87	84	0,965517	**448**	130	128	0,984615
149	90	94	1,044444	**449**	15	16	1,066667
150	25	24	0,96	**450**	64	65	1,015625
151	51	52	1,019608	**451**	36	35	0,972222
152	42	41	0,97619	**452**	55	53	0,963636
153	112	114	1,017857	**453**	27	28	1,037037
154	102	104	1,019608	**454**	91	95	1,043956
155	92	88	0,956522	**455**	38	37	0,973684
156	100	102	1,02	**456**	47	45	0,957447
157	117	113	0,965812	**457**	57	58	1,017544
158	38	39	1,026316	**458**	64	65	1,015625
159	60	59	0,983333	**459**	32	31	0,96875
160	46	47	1,021739	**460**	19	19	1

№ кадра	алгоритм	фактическое значение	относительная точность	№ кадра	алгоритм	фактическое значение	относительная точность
161	83	85	1,024096	**461**	26	27	1,038462
162	17	17	1	**462**	93	97	1,043011
163	43	44	1,023256	**463**	113	108	0,955752
164	124	120	0,967742	**464**	81	79	0,975309
165	38	39	1,026316	**465**	22	23	1,045455
166	101	98	0,970297	**466**	107	110	1,028037
167	16	17	1,0625	**467**	73	70	0,958904
168	131	136	1,038168	**468**	93	89	0,956989
169	109	107	0,981651	**469**	19	20	1,052632
170	72	74	1,027778	**470**	36	37	1,027778
171	22	21	0,954545	**471**	37	36	0,972973
172	114	116	1,017544	**472**	34	33	0,970588
173	16	16	1	**473**	67	69	1,029851
174	49	50	1,020408	**474**	78	80	1,025641
175	72	75	1,041667	**475**	107	102	0,953271
176	70	67	0,957143	**476**	127	122	0,96063
177	17	18	1,058824	**477**	70	72	1,028571
178	25	24	0,96	**478**	19	20	1,052632
179	14	15	1,071429	**479**	47	45	0,957447
180	134	131	0,977612	**480**	99	96	0,969697
181	97	100	1,030928	**481**	122	127	1,040984
182	50	51	1,02	**482**	39	40	1,025641
183	12	12	1	**483**	95	91	0,957895
184	53	55	1,037736	**484**	99	95	0,959596
185	111	109	0,981982	**485**	126	130	1,031746
186	122	127	1,040984	**486**	83	86	1,036145
187	122	120	0,983607	**487**	114	112	0,982456
188	107	111	1,037383	**488**	33	34	1,030303
189	71	73	1,028169	**489**	85	87	1,023529
190	46	44	0,956522	**490**	42	40	0,952381
191	43	44	1,023256	**491**	124	126	1,016129
192	94	90	0,957447	**492**	68	71	1,044118
193	121	124	1,024793	**493**	12	12	1
194	111	109	0,981982	**494**	37	38	1,027027
195	47	49	1,042553	**495**	20	21	1,05
196	74	76	1,027027	**496**	121	116	0,958678
197	109	105	0,963303	**497**	67	69	1,029851
198	121	123	1,016529	**498**	83	85	1,024096
199	63	62	0,984127	**499**	92	90	0,978261
200	27	28	1,037037	**500**	50	51	1,02
201	19	19	1	**501**	102	104	1,019608
202	14	15	1,071429	**502**	23	22	0,956522

№ кадра	алгоритм	фактическое значение	относительная точность	№ кадра	алгоритм	фактическое значение	относительная точность
203	73	75	1,027397	**503**	20	21	1,05
204	86	82	0,953488	**504**	60	59	0,983333
205	57	58	1,017544	**505**	81	83	1,024691
206	104	102	0,980769	**506**	66	64	0,969697
207	49	51	1,040816	**507**	131	133	1,015267
208	88	84	0,954545	**508**	49	47	0,959184
209	120	123	1,025	**509**	136	140	1,029412
210	19	20	1,052632	**510**	25	24	0,96
211	97	93	0,958763	**511**	65	67	1,030769
212	134	138	1,029851	**512**	66	63	0,954545
213	89	87	0,977528	**513**	99	102	1,030303
214	113	115	1,017699	**514**	33	32	0,969697
215	39	38	0,974359	**515**	99	102	1,030303
216	132	132	1	**516**	49	47	0,959184
217	64	66	1,03125	**517**	116	120	1,034483
218	105	101	0,961905	**518**	27	26	0,962963
219	49	50	1,020408	**519**	65	66	1,015385
220	20	20	1	**520**	61	59	0,967213
221	47	49	1,042553	**521**	30	31	1,033333
222	88	85	0,965909	**522**	97	94	0,969072
223	26	25	0,961538	**523**	118	121	1,025424
224	59	61	1,033898	**524**	48	46	0,958333
225	91	89	0,978022	**525**	41	42	1,02439
226	61	63	1,032787	**526**	25	24	0,96
227	65	64	0,984615	**527**	64	66	1,03125
228	50	51	1,02	**528**	20	20	1
229	51	50	0,980392	**529**	40	41	1,025
230	106	108	1,018868	**530**	41	40	0,97561
231	16	17	1,0625	**531**	38	39	1,026316
232	52	51	0,980769	**532**	79	76	0,962025
233	107	109	1,018692	**533**	32	33	1,03125
234	89	86	0,966292	**534**	46	44	0,956522
235	99	103	1,040404	**535**	93	96	1,032258
236	135	129	0,955556	**536**	112	107	0,955357
237	48	49	1,020833	**537**	47	49	1,042553
238	48	50	1,041667	**538**	78	76	0,974359
239	15	15	1	**539**	123	125	1,01626
240	85	87	1,023529	**540**	14	14	1
241	83	81	0,975904	**541**	77	80	1,038961
242	114	119	1,04386	**542**	22	21	0,954545
243	114	112	0,982456	**543**	73	76	1,041096
244	42	43	1,02381	**544**	88	85	0,965909

№ кадра	алгоритм	фактическое значение	относительная точность	№ кадра	алгоритм	фактическое значение	относительная точность
245	87	90	1,034483	**545**	52	54	1,038462
246	41	40	0,97561	**546**	85	82	0,964706
247	81	84	1,037037	**547**	98	102	1,040816
248	60	58	0,966667	**548**	101	97	0,960396
249	28	29	1,035714	**549**	66	68	1,030303
250	100	96	0,96	**550**	133	127	0,954887
251	98	95	0,969388	**551**	27	28	1,037037
252	124	128	1,032258	**552**	26	25	0,961538
253	26	25	0,961538	**553**	12	13	1,083333
254	130	132	1,015385	**554**	103	99	0,961165
255	39	38	0,974359	**555**	81	84	1,037037
256	42	43	1,02381	**556**	52	54	1,038462
257	62	61	0,983871	**557**	133	137	1,030075
258	95	94	0,989474	**558**	77	80	1,038961
259	106	109	1,028302	**559**	65	67	1,030769
260	108	104	0,962963	**560**	73	76	1,041096
261	117	119	1,017094	**561**	33	34	1,030303
262	127	121	0,952756	**562**	49	50	1,020408
263	31	32	1,032258	**563**	71	74	1,042254
264	110	105	0,954545	**564**	18	19	1,055556
265	111	106	0,954955	**565**	133	135	1,015038
266	50	51	1,02	**566**	55	57	1,036364
267	127	125	0,984252	**567**	12	13	1,083333
268	83	85	1,024096	**568**	24	25	1,041667
269	115	111	0,965217	**569**	107	109	1,018692
270	27	28	1,037037	**570**	104	108	1,038462
271	61	60	0,983607	**571**	78	81	1,038462
272	99	95	0,959596	**572**	67	70	1,044776
273	113	115	1,017699	**573**	73	75	1,027397
274	73	70	0,958904	**574**	76	78	1,026316
275	21	22	1,047619	**575**	102	106	1,039216
276	105	102	0,971429	**576**	126	130	1,031746
277	105	109	1,038095	**577**	111	113	1,018018
278	31	30	0,967742	**578**	73	75	1,027397
279	12	12	1	**579**	74	77	1,040541
280	116	118	1,017241	**580**	88	91	1,034091
281	37	36	0,972973	**581**	90	94	1,044444
282	74	77	1,040541	**582**	84	87	1,035714
283	92	90	0,978261	**583**	126	130	1,031746
284	109	113	1,036697	**584**	108	110	1,018519
285	69	66	0,956522	**585**	42	43	1,02381
286	32	33	1,03125	**586**	42	43	1,02381

№ кадра	алгоритм	фактическое значение	относительная точность	№ кадра	алгоритм	фактическое значение	относительная точность
287	104	106	1,019231	**587**	75	78	1,04
288	133	128	0,962406	**588**	119	121	1,016807
289	61	63	1,032787	**589**	35	36	1,028571
290	87	83	0,954023	**590**	109	111	1,018349
291	135	141	1,044444	**591**	97	100	1,030928
292	90	86	0,955556	**592**	71	73	1,028169
293	18	18	1	**593**	27	28	1,037037
294	115	120	1,043478	**594**	106	110	1,037736
295	28	27	0,964286	**595**	93	95	1,021505
296	80	83	1,0375	**596**	37	38	1,027027
297	121	119	0,983471	**597**	123	126	1,02439
298	86	88	1,023256	**598**	127	131	1,031496
299	119	116	0,97479	**599**	55	56	1,018182
300	121	118	0,975207	**600**	117	119	1,017094

Б.2 Результаты сравнения расчетных значений предложенной методики с ручным измерением

Таблица Б.2 – Результаты испытаний

№	расчетное значение	фактическое значение	относительная точность	№	расчетное значение	фактическое значение	относительная точность
1	32,74	31,01	0,947159	**471**	10,19	10,69	0,953227
2	33,88	35,95	0,94242	**472**	29,5	30,63	0,963108
3	5,97	5,79	0,969849	**473**	20,97	20,34	0,969957
4	27,39	28,71	0,954023	**474**	29,42	31,05	0,947504
5	30,36	29,21	0,962121	**475**	13,52	14,21	0,951443
6	21,24	22,41	0,947791	**476**	24,48	23,45	0,957925
7	18,26	18,21	0,997262	**477**	14,25	14,95	0,953177
8	11,75	11,2	0,953191	**478**	12,34	12,98	0,950693
9	35,82	37,5	0,9552	**479**	13,72	12,84	0,93586
10	16,27	15,44	0,948986	**480**	31,37	33,52	0,935859
11	32,57	33,82	0,96304	**481**	18,18	18,43	0,986435
12	14,26	13,48	0,945302	**482**	19,72	18,39	0,932556
13	25,38	26,6	0,954135	**483**	30,49	32,54	0,937001
14	17,33	17,21	0,993076	**484**	33,64	33,69	0,998516
15	30,59	29,15	0,952926	**485**	22,95	21,64	0,942919
16	11,61	12,09	0,960298	**486**	29,9	31,33	0,954357
17	31,96	30,2	0,944931	**487**	7,94	7,79	0,981108
18	19,06	20,03	0,951573	**488**	30,07	28,49	0,947456
19	35,88	34,17	0,952341	**489**	34,62	36,58	0,946419
20	34,33	36,33	0,944949	**490**	21,5	21,18	0,985116
21	5,27	5,31	0,992467	**491**	33,88	32,45	0,957792

№	расчетное значение	фактическое значение	относительная точность	№	расчетное значение	фактическое значение	относительная точность
22	17,86	17,19	0,962486	**492**	9,06	9,7	0,934021
23	14,2	15,06	0,942895	**493**	30,97	30,88	0,997094
24	21,38	20,06	0,93826	**494**	34,61	33,02	0,95406
25	35,17	37,48	0,938367	**495**	19,8	20,87	0,94873
26	15,01	14,23	0,948035	**496**	27,31	27,39	0,997079
27	34,02	35,58	0,956155	**497**	26,63	24,81	0,931656
28	20,17	20,25	0,996049	**498**	18,1	19	0,952632
29	7,24	6,78	0,936464	**499**	19,73	20,05	0,98404
30	32,55	34,27	0,94981	**500**	27,81	26,72	0,960805
31	7,76	7,34	0,945876	**501**	13,7	14,5	0,944828
32	29,65	30,83	0,961726	**502**	33,15	32,64	0,984615
33	6,1	5,77	0,945902	**503**	22,81	21,29	0,933363
34	25,43	26,82	0,948173	**504**	26,66	27,81	0,958648
35	13,52	13,53	0,999261	**505**	11,54	11,67	0,98886
36	24,21	22,7	0,937629	**506**	21,4	20,41	0,953738
37	5,37	5,69	0,943761	**507**	33,96	35,52	0,956081
38	30,03	28,83	0,96004	**508**	7,65	7,77	0,984556
39	33,27	34,78	0,956584	**509**	29,68	27,83	0,937668
40	14,44	13,74	0,951524	**510**	18,97	20,2	0,939109
41	23,81	24,95	0,954309	**511**	14,12	14,17	0,996471
42	9,66	9,61	0,994824	**512**	33,97	31,61	0,930527
43	20,34	19,16	0,941986	**513**	20,04	20,94	0,95702
44	13,29	13,96	0,952006	**514**	10,49	10,6	0,989623
45	28,86	27,91	0,967082	**515**	35,56	33,79	0,950225
46	15,86	16,58	0,956574	**516**	7,29	7,68	0,949219
47	15,97	15	0,939261	**517**	11,32	11,31	0,999117
48	35,66	37,34	0,955008	**518**	34,41	32,55	0,945946
49	33,71	33,05	0,980421	**519**	30,42	32,19	0,945014
50	26,11	24,32	0,931444	**520**	30,01	29,63	0,987338
51	8,4	8,98	0,935412	**521**	27,49	26,16	0,951619
52	5,64	5,31	0,941489	**522**	24,3	25,89	0,938586
53	5,52	5,8	0,951724	**523**	24,3	24,48	0,992647
54	14,51	13,81	0,951757	**524**	11,52	10,94	0,949653
55	33,47	35,24	0,949773	**525**	17,48	18,34	0,953108
56	7,42	7,41	0,998652	**526**	29,81	30,19	0,987413
57	28,33	26,73	0,943523	**527**	14,24	13,32	0,935393
58	10,18	10,76	0,946097	**528**	16,45	17,18	0,957509
59	18,22	17,44	0,95719	**529**	13,99	13,95	0,997141
60	29,43	30,94	0,951196	**530**	11,22	10,48	0,934046
61	16,32	15,83	0,969975	**531**	25,83	27,44	0,941327
62	6	6,34	0,946372	**532**	23,23	23,28	0,997852
63	24,42	24,53	0,995516	**533**	18,31	17,52	0,956854

№	расчетное значение	фактическое значение	относительная точность	№	расчетное значение	фактическое значение	относительная точность
64	25,45	23,72	0,932024	**534**	32,82	34,78	0,943646
65	9,81	10,46	0,937859	**535**	23,59	23,56	0,998728
66	35,38	34,28	0,968909	**536**	7,26	7,04	0,969697
67	24,09	25,74	0,935897	**537**	29,27	31,04	0,942977
68	22,44	21,27	0,947861	**538**	23,43	23,22	0,991037
69	15,82	16,94	0,933884	**539**	15,57	14,65	0,940912
70	29,42	29,27	0,994901	**540**	23,55	24,85	0,947686
71	26,45	24,92	0,942155	**541**	33,52	33,29	0,993138
72	25,26	27,01	0,935209	**542**	11,42	10,63	0,930823
73	25,98	24,85	0,956505	**543**	7,84	8,35	0,938922
74	22,14	23,39	0,946558	**544**	28,34	27,8	0,980946
75	5,96	5,71	0,958054	**545**	33,04	31,18	0,943705
76	35,09	36,54	0,960317	**546**	26,89	28,72	0,936281
77	25,07	24,81	0,989629	**547**	33,35	32,27	0,967616
78	11,2	10,81	0,965179	**548**	7,08	7,48	0,946524
79	30,34	32,2	0,942236	**549**	36,06	33,99	0,942596
80	27,78	26,46	0,952484	**550**	20,06	21,31	0,941342
81	27,86	29,58	0,941853	**551**	15,28	14,34	0,938482
82	22,22	21,52	0,968497	**552**	35,61	37,27	0,95546
83	27,85	29,62	0,940243	**553**	9,63	9,08	0,942887
84	19,34	19,39	0,997421	**554**	14,09	14,9	0,945638
85	24,23	22,55	0,930664	**555**	25,92	24,14	0,931327
86	8,35	8,89	0,939258	**556**	14,73	15,4	0,956494
87	23,4	22,65	0,967949	**557**	8,51	8,24	0,968273
88	5,09	5,33	0,954972	**558**	24,7	26,35	0,937381
89	24,92	23,62	0,947833	**559**	15,6	14,72	0,94359
90	23,19	24,2	0,958264	**560**	8,26	8,64	0,956019
91	9,24	9,26	0,99784	**561**	14,45	13,66	0,945329
92	21,14	19,87	0,939924	**562**	9,02	9,39	0,960596
93	29,42	31,19	0,943251	**563**	9,32	8,92	0,957082
94	8,19	7,9	0,964591	**564**	11,01	11,46	0,960733
95	5,02	5,34	0,940075	**565**	19,06	17,82	0,934942
96	30,77	28,82	0,936627	**566**	7,63	7,92	0,963384
97	13,42	14,22	0,943741	**567**	14,45	13,54	0,937024
98	9,48	9,47	0,998945	**568**	5,39	5,6	0,9625
99	25,81	24,94	0,966292	**569**	5,81	5,48	0,943201
100	20,09	21,29	0,943636	**570**	12,86	13,39	0,960418
101	23,34	21,79	0,93359	**571**	11,68	11,15	0,954623
102	26,83	28,35	0,946384	**572**	26,26	27,68	0,948699
103	28,09	27,12	0,965468	**573**	19,73	18,57	0,941206
104	18,43	19,34	0,952947	**574**	8,55	8,95	0,955307
105	16,28	16,03	0,984644	**575**	33,01	30,9	0,93608

№	расчетное значение	фактическое значение	относительная точность	№	расчетное значение	фактическое значение	относительная точность
106	34,8	33,04	0,949425	**576**	18,21	18,98	0,959431
107	35,98	37,5	0,959467	**577**	25,77	24,92	0,967016
108	24,71	23,54	0,952651	**578**	26,18	27,82	0,94105
109	35,83	37,9	0,945383	**579**	22,37	20,85	0,932052
110	8,72	8,42	0,965596	**580**	28,55	29,65	0,962901
111	30,85	32,29	0,955404	**581**	15,54	14,45	0,929858
112	7,79	7,91	0,984829	**582**	20,07	21,04	0,953897
113	27,21	25,84	0,949651	**583**	21,46	20,4	0,950606
114	23,89	25,49	0,93723	**584**	32,46	34,34	0,945253
115	10,16	9,78	0,962598	**585**	18,35	17,63	0,960763
116	15,12	15,92	0,949749	**586**	32,05	34,12	0,939332
117	7,86	7,43	0,945293	**587**	15,15	14,47	0,955116
118	20,13	21,41	0,940215	**588**	22,91	23,89	0,958979
119	15,71	15,75	0,99746	**589**	33,48	31,78	0,949223
120	18,48	17,68	0,95671	**590**	6,65	7,11	0,935302
121	14,98	16,04	0,933915	**591**	8,92	8,47	0,949552
122	24,15	23,19	0,960248	**592**	28,64	30,27	0,946151
123	20,65	21,62	0,955134	**593**	28,61	27,6	0,964698
124	16,43	15,78	0,960438	**594**	29,07	30,25	0,960992
125	20,7	21,89	0,945637	**595**	29,12	27,44	0,942308
126	13,84	13,62	0,984104	**596**	29,49	31,01	0,950984
127	27	25,82	0,956296	**597**	6,94	6,68	0,962536
128	7,96	8,33	0,955582	**598**	19,45	20,8	0,935096
129	25,94	24,4	0,940632	**599**	31,99	30,22	0,94467
130	28,59	30,24	0,945437	**600**	11,23	11,71	0,959009
131	11,49	11,14	0,969539	**601**	28,05	27,13	0,967201
132	31,72	33,54	0,945736	**602**	6,38	6,63	0,962293
133	22,62	21,27	0,940318	**603**	23,7	22,7	0,957806
134	33,49	35,4	0,946045	**604**	17,64	18,85	0,935809
135	23,11	22,03	0,953267	**605**	10,61	9,95	0,937795
136	21,08	22,33	0,944021	**606**	8,2	8,58	0,955711
137	13,89	13,34	0,960403	**607**	35,86	33,7	0,939766
138	8	8,47	0,94451	**608**	36,18	38,04	0,951104
139	31,48	29,44	0,935197	**609**	14,08	13,51	0,959517
140	13,92	14,73	0,94501	**610**	20,86	22,17	0,940911
141	7,56	7,13	0,943122	**611**	16,96	15,86	0,935142
142	6,35	6,67	0,952024	**612**	12,08	12,86	0,939347
143	34,61	32,43	0,937012	**613**	30,94	29,06	0,939237
144	24,84	26,1	0,951724	**614**	29,79	31,1	0,957878
145	19,31	18,16	0,940445	**615**	20,7	19,35	0,934783
146	30,25	31,8	0,951258	**616**	23,44	24,66	0,950527
147	31,69	30,04	0,947933	**617**	25,06	23,8	0,949721

№	расчетное значение	фактическое значение	относительная точность	№	расчетное значение	фактическое значение	относительная точность
148	17,19	18,4	0,934239	**618**	19,48	20,51	0,949781
149	19,09	18,11	0,948664	**619**	15,57	14,48	0,929994
150	26,16	28,01	0,933952	**620**	29,83	31,69	0,941306
151	34,69	32,48	0,936293	**621**	17,49	16,28	0,930818
152	9,44	9,98	0,945892	**622**	31,14	32,72	0,951711
153	20,17	18,87	0,935548	**623**	15,28	14,65	0,95877
154	33,26	34,99	0,950557	**624**	7,21	6,99	0,969487
155	20,25	19,04	0,940247	**625**	28,32	27,17	0,959393
156	32,62	34,76	0,938435	**626**	9,9	9,39	0,948485
157	26,99	25,62	0,94924	**627**	17,1	16,12	0,94269
158	22,6	23,82	0,948783	**628**	28,67	27,19	0,948378
159	14,58	13,78	0,94513	**629**	11,27	10,73	0,952085
160	24,74	26,48	0,93429	**630**	20,73	19,45	0,938254
161	7,77	7,33	0,943372	**631**	20,35	19,35	0,95086
162	29,5	31,25	0,944	**632**	20,48	19,39	0,946777
163	7,71	7,27	0,942931	**633**	34,51	32,91	0,953637
164	9,73	10,4	0,935577	**634**	8,93	8,3	0,929451
165	35,4	33,92	0,958192	**635**	21,98	20,79	0,94586
166	35,08	36,74	0,954818	**636**	33,6	32,49	0,966964
167	19,29	18,67	0,967859	**637**	33,65	31,86	0,946805
168	6,98	7,37	0,947083	**638**	5,76	5,57	0,967014
169	17	16,05	0,944118	**639**	9,37	8,92	0,951974
170	12,2	12,73	0,958366	**640**	29,05	27,2	0,936317
171	31,41	30,19	0,961159	**641**	31,03	29,96	0,965517
172	10,53	11,2	0,940179	**642**	33,67	31,8	0,944461
173	19,97	19,25	0,963946	**643**	27,78	26,87	0,967243
174	8,51	9,06	0,939294	**644**	32,86	31,78	0,967133
175	21,68	20,55	0,947878	**645**	21,6	20,81	0,963426
176	34,87	37,26	0,935856	**646**	20,13	18,71	0,929459
177	24,51	23,21	0,94696	**647**	16,86	15,7	0,931198
178	14,49	15,5	0,934839	**648**	7,94	7,55	0,950882
179	24,66	23,84	0,966748	**649**	16,67	16,03	0,961608
180	22,01	23,04	0,955295	**650**	17,74	16,97	0,956595
181	22,65	21,31	0,940839	**651**	30,82	29,18	0,946788
182	22,16	23,65	0,936998	**652**	33,41	31,89	0,954505
183	31,43	29,86	0,950048	**653**	16,22	15,55	0,958693
184	29,33	30,53	0,960694	**654**	5,28	5	0,94697
185	26,16	25,02	0,956422	**655**	25,18	24,35	0,967037
186	8,78	9,29	0,945102	**656**	31,5	29,57	0,93873
187	29,37	27,97	0,952332	**657**	23,07	21,48	0,931079
188	5,88	6,27	0,937799	**658**	23,52	22,7	0,965136
189	11,78	11,1	0,942275	**659**	29,18	27,79	0,952365

№	расчетное значение	фактическое значение	относительная точность	№	расчетное значение	фактическое значение	относительная точность
190	9,92	10,36	0,957529	**660**	8,46	7,87	0,93026
191	27,83	26,84	0,964427	**661**	13,97	13,53	0,968504
192	11,92	12,68	0,940063	**662**	29,72	28,6	0,962315
193	20,21	19,25	0,952499	**663**	35,71	34,46	0,964996
194	7,57	8,04	0,941542	**664**	9,84	9,45	0,960366
195	30,19	28,88	0,956608	**665**	31,23	29,29	0,93788
196	14,69	15,26	0,962647	**666**	9,01	8,47	0,940067
197	31,73	30,63	0,965332	**667**	23,7	22,17	0,935443
198	9,03	9,44	0,956568	**668**	22,58	21,27	0,941984
199	14,18	13,18	0,929478	**669**	14,28	13,65	0,955882
200	10,42	11,03	0,944696	**670**	12,79	12,14	0,949179
201	16,55	15,63	0,944411	**671**	8,47	8,03	0,948052
202	25,59	27,23	0,939772	**672**	27,74	26,64	0,960346
203	8,47	8,06	0,951594	**673**	9	8,53	0,947778
204	5,76	6,05	0,952066	**674**	32,56	31,27	0,960381
205	21,33	20,54	0,962963	**675**	5,86	5,64	0,962457
206	21,67	23,05	0,94013	**676**	34,05	32,8	0,963289
207	15,38	14,62	0,950585	**677**	19,62	18,75	0,955657
208	20,22	21,52	0,939591	**678**	11,52	10,89	0,945313
209	21,92	20,57	0,938412	**679**	34,33	32,07	0,934168
210	7,48	7,87	0,950445	**680**	21,49	20,82	0,968823
211	29,54	27,69	0,937373	**681**	12,35	11,52	0,932794
212	9,46	10,06	0,940358	**682**	25,64	24,8	0,967239
213	10,4	9,68	0,930769	**683**	16,46	15,77	0,95808
214	19,32	20,07	0,962631	**684**	14,42	13,44	0,932039
215	29,73	27,96	0,940464	**685**	22,5	21,39	0,950667
216	33,26	34,53	0,96322	**686**	10,12	9,57	0,945652
217	9,95	9,48	0,952764	**687**	27,92	26,72	0,95702
218	5,3	5,51	0,961887	**688**	5,56	5,37	0,965827
219	25,42	24,33	0,95712	**689**	26,7	25,11	0,940449
220	27,24	29,16	0,934156	**690**	33,15	31,68	0,955656
221	5,28	5	0,94697	**691**	6,15	5,94	0,965854
222	17,72	18,97	0,934106	**692**	13,3	12,58	0,945865
223	27,9	26,59	0,953047	**693**	8,47	8,1	0,956316
224	11,92	12,69	0,939322	**694**	7,04	6,63	0,941761
225	33,92	31,77	0,936616	**695**	10,25	9,7	0,946341
226	9,33	9,72	0,959877	**696**	17,98	17,4	0,967742
227	24,85	23,39	0,941247	**697**	11,48	11,11	0,96777
228	23,46	24,48	0,958333	**698**	30,56	28,63	0,936846
229	16,18	15,64	0,966625	**699**	27,01	25,69	0,951129
230	5,17	5,46	0,946886	**700**	5,32	5,12	0,962406
231	33,49	32,05	0,957002	**701**	19,56	18,94	0,968303

№	расчетное значение	фактическое значение	относительная точность	№	расчетное значение	фактическое значение	относительная точность
232	8,9	9,36	0,950855	**702**	36,03	34,62	0,960866
233	16,6	15,45	0,930723	**703**	23,34	22,07	0,945587
234	21,9	22,83	0,959264	**704**	13,69	12,75	0,931337
235	18,42	17,62	0,956569	**705**	20,57	19,68	0,956733
236	15,16	16,07	0,943373	**706**	22,54	21,52	0,954747
237	17,17	16,64	0,969132	**707**	25,73	24,83	0,965021
238	9,3	9,77	0,951894	**708**	29,47	28,34	0,961656
239	24,79	23,51	0,948366	**709**	5,01	4,66	0,93014
240	34,43	36,26	0,949531	**710**	16,29	15,44	0,947821
241	28,95	27,94	0,965112	**711**	10,73	10,4	0,969245
242	16,84	17,82	0,945006	**712**	12,68	11,8	0,930599
243	29,51	28,01	0,94917	**713**	6,93	6,53	0,94228
244	24,3	25,38	0,957447	**714**	26,45	25,02	0,945936
245	21,4	20,24	0,945794	**715**	28,02	26,41	0,942541
246	32,46	34,21	0,948845	**716**	18,35	17,5	0,953678
247	15,81	15,28	0,966477	**717**	17,01	16,18	0,951205
248	15,27	16,13	0,946683	**718**	34,02	32,86	0,965902
249	35,07	32,65	0,930995	**719**	34,54	32,54	0,942096
250	6,9	7,27	0,949106	**720**	28	27,08	0,967143
251	18,51	17,28	0,933549	**721**	8,51	8,1	0,951821
252	36,15	34,38	0,951037	**722**	14,67	14,18	0,966599
253	19,13	20,2	0,94703	**723**	28,35	26,81	0,945679
254	27,28	26,44	0,969208	**724**	22,57	21,77	0,964555
255	35,92	37,82	0,949762	**725**	28,75	27,14	0,944
256	11,15	10,69	0,958744	**726**	30,63	28,67	0,93601
257	35,6	33,54	0,942135	**727**	33,94	32,8	0,966411
258	23,45	24,9	0,941767	**728**	18,85	17,62	0,934748
259	12,51	11,81	0,944045	**729**	19,4	18,43	0,95
260	6,09	6,34	0,960568	**730**	18,22	17,47	0,958836
261	22,34	20,94	0,937332	**731**	30,43	29,19	0,959251
262	31,63	30,32	0,958584	**732**	25,31	23,52	0,929277
263	16,65	17,4	0,956897	**733**	34,56	33,18	0,960069
264	12,37	11,79	0,953112	**734**	33,41	31,11	0,931158
265	12,97	13,78	0,941219	**735**	34,26	32,39	0,945417
266	34,15	31,75	0,929722	**736**	30,69	29,67	0,966764
267	32,11	30,24	0,941763	**737**	30,78	28,81	0,935997
268	15,09	15,81	0,954459	**738**	16,68	15,51	0,929856
269	10,67	10,09	0,945642	**739**	31,07	29,2	0,939813
270	35,73	37,73	0,946992	**740**	25,9	25,04	0,966795
271	32,35	30,85	0,953632	**741**	31,54	30,33	0,961636
272	19,58	18,35	0,937181	**742**	15,15	14,42	0,951815
273	28,08	29,49	0,952187	**743**	19,72	18,98	0,962475

№	расчетное значение	фактическое значение	относительная точность	№	расчетное значение	фактическое значение	относительная точность
274	18,56	17,25	0,929418	**744**	23,03	22,17	0,962657
275	17,23	18,32	0,940502	**745**	24,4	22,94	0,940164
276	6,55	6,15	0,938931	**746**	23,6	22,32	0,945763
277	13,81	13,05	0,944967	**747**	14,39	13,78	0,957609
278	24,46	25,76	0,949534	**748**	6,88	6,43	0,934593
279	22,46	20,89	0,930098	**749**	16,18	15,62	0,965389
280	11,38	12,02	0,946755	**750**	29,78	28,51	0,957354
281	8,58	8,08	0,941725	**751**	22,85	21,31	0,932604
282	34,01	32,9	0,967363	**752**	25,81	24	0,929872
283	9,22	9,71	0,949537	**753**	35,83	34,28	0,95674
284	14,1	13,65	0,968085	**754**	8,13	7,8	0,95941
285	8,02	8,55	0,938012	**755**	32,34	31,35	0,969388
286	16,68	16,03	0,961031	**756**	18,16	17,23	0,948789
287	11,7	11,03	0,942735	**757**	25,82	24,11	0,933772
288	16,17	17,19	0,940663	**758**	14,85	14,28	0,961616
289	9,55	9,08	0,950785	**759**	25,53	23,87	0,934978
290	5,68	6,06	0,937294	**760**	10,05	9,69	0,964179
291	29,55	28,03	0,948562	**761**	20,92	20,04	0,957935
292	30,97	28,91	0,933484	**762**	22,62	21,5	0,950486
293	9,67	10,18	0,949902	**763**	7,68	7,25	0,94401
294	5,01	4,83	0,964072	**764**	21,53	20,36	0,945657
295	29,26	30,44	0,961235	**765**	22,86	21,66	0,947507
296	13,44	12,73	0,947173	**766**	18,48	17,58	0,951299
297	35,96	33,51	0,931869	**767**	19,32	18,1	0,936853
298	7,68	8,09	0,94932	**768**	15,52	14,73	0,949098
299	26,59	24,96	0,938699	**769**	29,77	27,77	0,932818
300	23,64	25	0,9456	**770**	24,36	23,28	0,955665
301	25,64	24,39	0,951248	**771**	26,31	25,35	0,963512
302	18,12	17,29	0,954194	**772**	12,92	12,4	0,959752
303	26,09	27,43	0,951148	**773**	21,03	20,02	0,951973
304	22,77	21,3	0,935441	**774**	25,75	24,74	0,960777
305	7,81	8,11	0,963009	**775**	12,84	12,03	0,936916
306	6,33	6,04	0,954186	**776**	35,87	34,41	0,959297
307	30,11	28,59	0,949518	**777**	26,57	25,38	0,955213
308	8,54	9,14	0,934354	**778**	10,27	9,77	0,951315
309	27,25	25,8	0,946789	**779**	23,65	22,29	0,942495
310	15,13	16,08	0,94092	**780**	24,19	23,28	0,962381
311	24	22,76	0,948333	**781**	36,09	34,19	0,947354
312	14,77	14,31	0,968856	**782**	9,38	8,85	0,943497
313	11,06	11,71	0,944492	**783**	25,98	24,44	0,940724
314	28,89	27,28	0,944271	**784**	14,57	13,79	0,946465
315	33,08	34,65	0,95469	**785**	19,55	18,32	0,937084

№	расчетное значение	фактическое значение	относительная точность	№	расчетное значение	фактическое значение	относительная точность
316	9,84	9,37	0,952236	**786**	30,03	28,49	0,948718
317	27,21	25,82	0,948916	**787**	6,16	5,94	0,964286
318	23,38	24,49	0,954675	**788**	5,28	5,1	0,965909
319	33,43	31,25	0,934789	**789**	20,52	19,88	0,968811
320	35,5	37,97	0,934949	**790**	24,61	23,09	0,938236
321	18,72	17,95	0,958868	**791**	23,43	22,55	0,962441
322	6,95	6,56	0,943885	**792**	10,26	10,97	0,935278
323	29,17	30,99	0,941271	**793**	14,3	14,91	0,959088
324	16,88	15,86	0,939573	**794**	32,97	34,65	0,951515
325	17,81	18,58	0,958558	**795**	9,26	9,9	0,935354
326	29,88	28,97	0,969545	**796**	34,66	36,98	0,937263
327	19,87	19,2	0,966281	**797**	33,03	35,05	0,942368
328	26,08	27,33	0,954263	**798**	18,29	19,11	0,957091
329	35,21	33,46	0,950298	**799**	19,41	20,5	0,946829
330	33,78	35,07	0,963216	**800**	16,01	16,69	0,959257
331	29,06	27,92	0,960771	**801**	25,77	27,14	0,949521
332	9,88	9,22	0,933198	**802**	10,02	10,5	0,954286
333	22,04	23,46	0,939471	**803**	35,83	38,37	0,933802
334	34,31	32,34	0,942582	**804**	22,96	24,23	0,947586
335	13,92	14,46	0,962656	**805**	32,02	33,54	0,954681
336	35,65	33,94	0,952034	**806**	5,26	5,46	0,96337
337	28,6	27,08	0,946853	**807**	15,58	16,23	0,959951
338	33,75	35,33	0,955279	**808**	28,79	30,12	0,955843
339	21,54	20,57	0,954968	**809**	18,81	19,99	0,94097
340	7,81	8,18	0,954768	**810**	28,31	29,92	0,94619
341	15,72	15,04	0,956743	**811**	25,6	27,39	0,934648
342	29,16	27,75	0,951646	**812**	8,28	8,72	0,949541
343	6,14	6,38	0,962382	**813**	24,92	25,94	0,960678
344	8,74	8,28	0,947368	**814**	26,84	27,92	0,961318
345	34,59	36,78	0,940457	**815**	24,23	25,65	0,944639
346	35,72	34,16	0,956327	**816**	22,05	23,54	0,936703
347	29,41	27,51	0,935396	**817**	11,07	11,65	0,950215
348	19,44	20,55	0,945985	**818**	25,5	27,02	0,943745
349	20,53	19,39	0,944472	**819**	34,82	36,44	0,955543
350	18,02	18,96	0,950422	**820**	10,28	10,7	0,960748
351	30,7	29,17	0,950163	**821**	31	32,49	0,95414
352	8,6	9,06	0,949227	**822**	19,98	20,87	0,957355
353	9,71	9,06	0,933059	**823**	35,22	37,66	0,93521
354	10,54	11,25	0,936889	**824**	22,14	23,38	0,946963
355	29,14	27,32	0,937543	**825**	5,56	5,92	0,939189
356	23,11	24,32	0,950247	**826**	8,71	9,28	0,938578
357	31,52	29,57	0,938135	**827**	35,34	37,38	0,945425

№	расчетное значение	фактическое значение	относительная точность	№	расчетное значение	фактическое значение	относительная точность
358	15,51	16,1	0,963354	**828**	28,74	30,36	0,94664
359	32,61	31,59	0,968721	**829**	29,75	30,95	0,961228
360	9,59	10,18	0,942043	**830**	11,75	12,23	0,960752
361	16,2	17,29	0,936958	**831**	10,13	10,63	0,952963
362	10,81	10,05	0,929695	**832**	14,19	15,19	0,934167
363	29,75	31,23	0,95261	**833**	18,43	19,26	0,956906
364	31,48	33,07	0,95192	**834**	20,78	22,11	0,939846
365	18,68	17,58	0,941113	**835**	31,57	33,64	0,938466
366	31,4	32,62	0,9626	**836**	6,65	7,1	0,93662
367	14,86	15,61	0,951954	**837**	22,91	23,83	0,961393
368	31,15	29,59	0,94992	**838**	30,8	32,13	0,958606
369	20,4	21,79	0,936209	**839**	28,78	30,46	0,944846
370	29,63	31,08	0,953346	**840**	18,99	19,79	0,959576
371	24,79	23,91	0,964502	**841**	10,99	11,56	0,950692
372	13,6	14,41	0,943789	**842**	10,85	11,51	0,942659
373	19,89	21,23	0,936882	**843**	33,8	35,44	0,953725
374	11,94	11,38	0,953099	**844**	34,55	36,27	0,952578
375	24,17	25,85	0,93501	**845**	22,19	23,21	0,956053
376	28,09	30,04	0,935087	**846**	28,22	30,19	0,934747
377	32,72	30,97	0,946516	**847**	30,61	32,52	0,941267
378	28,28	29,82	0,948357	**848**	13,65	14,51	0,940731
379	28,32	30,12	0,940239	**849**	32,29	34,29	0,941674
380	34,64	33,18	0,957852	**850**	7,61	7,92	0,960859
381	14,09	14,74	0,955902	**851**	24,42	25,92	0,94213
382	20,02	20,87	0,959272	**852**	12,8	13,69	0,934989
383	11,73	11,34	0,966752	**853**	26,5	28,34	0,935074
384	17,66	18,38	0,960827	**854**	12,52	13,25	0,944906
385	19,44	20,64	0,94186	**855**	5,54	5,9	0,938983
386	7,57	7,12	0,940555	**856**	12,52	13,26	0,944193
387	21,33	22,48	0,948843	**857**	25,38	26,93	0,942443
388	27,59	29,33	0,940675	**858**	13,72	14,48	0,947514
389	27,03	25,16	0,930818	**859**	11,32	11,97	0,945698
390	16,27	16,91	0,962153	**860**	21,08	22,16	0,951264
391	21,94	22,92	0,957243	**861**	6,16	6,5	0,947692
392	7,75	7,46	0,962581	**862**	33,03	34,86	0,947504
393	27,33	29,14	0,937886	**863**	11,1	11,77	0,943076
394	19,68	20,56	0,957198	**864**	30,19	31,94	0,94521
395	28,48	27,46	0,964185	**865**	12,56	13,04	0,96319
396	19,02	20,12	0,945328	**866**	10,02	10,61	0,944392
397	30,38	32,41	0,937365	**867**	6,64	7,02	0,945869
398	5,15	4,99	0,968932	**868**	32,69	34,4	0,950291
399	30,7	32,7	0,938838	**869**	35,34	37,64	0,938895

№	расчетное значение	фактическое значение	относительная точность	№	расчетное значение	фактическое значение	относительная точность
400	9,65	10,08	0,957341	**870**	9,38	10	0,938
401	8,43	8,03	0,95255	**871**	24,89	26,1	0,95364
402	24,51	25,85	0,948162	**872**	33,51	34,98	0,957976
403	26,21	27,49	0,953438	**873**	22,38	23,88	0,937186
404	11,18	10,49	0,938283	**874**	14,19	14,99	0,946631
405	13,94	14,48	0,962707	**875**	20,05	21,36	0,93867
406	14,06	14,75	0,95322	**876**	27,8	29,01	0,95829
407	8,93	8,54	0,956327	**877**	5,88	6,25	0,9408
408	28,38	29,86	0,950435	**878**	25,18	26,47	0,951266
409	18,09	18,93	0,955626	**879**	26,28	27,4	0,959124
410	23,52	22,53	0,957908	**880**	18,08	19	0,951579
411	28,24	30,11	0,937894	**881**	35,39	37,15	0,952624
412	25,54	27,15	0,9407	**882**	27,14	28,28	0,959689
413	23,03	22,24	0,965697	**883**	35,48	37,56	0,944622
414	6,14	6,5	0,944615	**884**	28,53	30,03	0,95005
415	8,01	8,49	0,943463	**885**	7,41	7,71	0,961089
416	13,47	12,95	0,961396	**886**	20,12	20,91	0,962219
417	30,82	32,7	0,942508	**887**	17,65	18,71	0,943346
418	12,48	13,26	0,941176	**888**	21,65	22,64	0,956272
419	26,6	25,1	0,943609	**889**	32,52	33,84	0,960993
420	34,51	35,83	0,963159	**890**	31,41	33,13	0,948083
421	18,63	19,95	0,933835	**891**	18,06	19,07	0,947037
422	6,59	6,25	0,948407	**892**	21,41	22,25	0,962247
423	14,61	15,64	0,934143	**893**	9	9,63	0,934579
424	25,55	26,75	0,95514	**894**	6,04	6,38	0,946708
425	26,9	25,84	0,960595	**895**	15,67	16,65	0,941141
426	18,85	19,82	0,95106	**896**	12,74	13,33	0,955739
427	18,99	20,04	0,947605	**897**	19,99	21,3	0,938498
428	35,71	34,6	0,968916	**898**	21	22,05	0,952381
429	32,97	35,11	0,939049	**899**	11,39	12,17	0,935908
430	29,51	30,79	0,958428	**900**	20,85	22,32	0,93414
431	22,19	21,23	0,956737	**901**	12,46	13,16	0,946809
432	23,84	25,4	0,938583	**902**	19,23	20,18	0,952924
433	22,98	24,44	0,940262	**903**	24,78	26,15	0,94761
434	31,07	29,31	0,943354	**904**	30,77	31,96	0,962766
435	16,78	17,61	0,952868	**905**	9,5	9,93	0,956697
436	29,74	31,52	0,943528	**906**	19,3	20,58	0,937804
437	6,68	6,28	0,94012	**907**	6,9	7,29	0,946502
438	15,85	16,53	0,958863	**908**	7,98	8,54	0,934426
439	34,46	35,93	0,959087	**909**	23,35	24,3	0,960905
440	15,71	14,85	0,945258	**910**	21,02	22,22	0,945995
441	21,92	23,05	0,950976	**911**	21,02	21,84	0,962454

№	расчетное значение	фактическое значение	относительная точность	№	расчетное значение	фактическое значение	относительная точность
442	19,88	21,06	0,94397	**912**	17,43	18,61	0,936593
443	30,93	29,98	0,969285	**913**	12,57	13,12	0,958079
444	22,07	23,29	0,947617	**914**	32,05	33,35	0,961019
445	28,25	30,1	0,938538	**915**	32,92	35,22	0,934696
446	17,75	17,14	0,965634	**916**	13,76	14,65	0,939249
447	29,55	31,53	0,937203	**917**	27,71	28,81	0,961819
448	21,79	23,28	0,935997	**918**	23,85	25,15	0,94831
449	18,67	17,59	0,942153	**919**	21,32	22,38	0,952636
450	34,91	36,37	0,959857	**920**	32,61	34,73	0,938958
451	35,77	37,74	0,947801	**921**	34,87	37,34	0,933851
452	21,29	20,58	0,966651	**922**	25,96	27,27	0,951962
453	35,51	37,68	0,94241	**923**	6,67	7,01	0,951498
454	31,65	32,86	0,963177	**924**	31,5	33,39	0,943396
455	5,51	5,34	0,969147	**925**	16,72	17,81	0,938798
456	10,84	11,28	0,960993	**926**	23,38	24,33	0,960954
457	14,44	15,22	0,948752	**927**	29,46	30,91	0,95309
458	19,69	18,47	0,93804	**928**	33,73	35,47	0,950944
459	33,04	35,15	0,939972	**929**	11,57	12,06	0,95937
460	35,18	37,36	0,941649	**930**	12,09	12,66	0,954976
461	35,79	34,31	0,958648	**931**	26,93	28,07	0,959387
462	33,57	35,88	0,935619	**932**	35,67	37,51	0,950946
463	33,61	35,47	0,947561	**933**	13,49	14,14	0,954031
464	13,1	12,18	0,929771	**934**	5,01	5,35	0,936449
465	28,93	30,65	0,943883	**935**	18,42	19,3	0,954404
466	5,33	5,64	0,945035	**936**	18,65	19,37	0,962829
467	7,39	6,92	0,936401	**937**	9,03	9,52	0,948529
468	16,25	17,07	0,951963	**938**	7,79	8,32	0,936298
469	14,01	14,66	0,955662	**939**	5,14	5,4	0,951852
470	7,58	7,25	0,956464	**940**	14,87	15,9	0,93522

ПРИЛОЖЕНИЕ В

Программа для ЭВМ

FoRest

Свидетельство о регистрации № 2014611152 от 27.01.2014

Название программы для ЭВМ: «FoRest»

Реферат: Программа предназначена для обработки и анализа фотоизображений штабелей круглого леса с целью определения его кубатуры. Программа на основе фотоизображений с осуществлением минимума ручных измерительных операций определяет объем каждого бревна в штабеле с учетом его пространственного положения и с высокой точностью вычисляет кубатуру круглого леса в штабеле. Программа обеспечивает выполнение следующих функций: анализ фотоизображений и их преобразование; построение математической трехмерной модели штабеля; распознавание каждого бревна в штабеле и его пространственной ориентации; вычисление объема каждого бревна и кубатуры штабеля; составление отчетов с индивидуальной информацией по каждому штабелю в форматах pdf, doc, xln.

Тип реализующей ЭВМ: IBM PC - совмест. ПК

Язык программирования: C++/CLI

Вид и версия операционной системы: Windows 7 Professional

Объем программы для ЭВМ: 13,2 Мб

Программный код:

Forest.cpp

```
// Forest.cpp: главный файл проекта.
#include "stdafx.h"
#include "MainForm.h"
#include "Model.h"
#include "MainPresenter.h"
using namespace Forest;
[STAThreadAttribute]
int main(array<System::String ^> ^args)
{
```

```
	// Включение визуальных эффектов Windows XP до создания каких-либо элементов
управления
	Application::EnableVisualStyles();
	Application::SetCompatibleTextRenderingDefault(false);
	try
	{
		/*создание главного окна*/
		IMainView^ mainView = gcnew MainForm();
		/*создание модели*/
		IModel^ model = gcnew Model();
		/*создание презенторов*/
		MainPresenter^ mainPresenter = gcnew MainPresenter(mainView,model);
		//запуск главного окна
		Application::Run(dynamic_cast<MainForm^>(mainView));
	}
	catch(Exception^ e)
	{
		MessageBox::Show("Ошибка! Приложение будет закрыто");
	}
	return 0;
}
```

MainForm.h

```
#pragma once
#include "IMainView.h"
#include "NavigatorForm.h"
#include "SettingsForm.h"
#include "ReportForm.h"
#include "Plot3DForm.h"
#include "ToolState.h"
#include "MainPresenter.h"
#include "MainFormSettings.h"
#include "IMainViewTools.h"
#include "AboutForm.h"
namespace Forest {
	using namespace System;
	using namespace System::ComponentModel;
	using namespace System::Collections;
	using namespace System::Windows::Forms;
	using namespace System::Data;
	using namespace System::Drawing;
	using namespace System::Drawing::Drawing2D;
	using namespace Global;
	using namespace Forest::Tools;
	using namespace AppRequest::UserRequest;
	using namespace System::Reflection;
	using namespace Forest::SavedSettings;
	using namespace Graphics2D;
	/// <summary>
	/// Сводка для Form1
	/// </summary>
ref class MainForm : public System::Windows::Forms::Form ,public IMainView, public
IMainViewTools
{
public:
	MainForm(void)
	{
		InitializeComponent();
		//
```

```
            //TODO: добавьте код конструктора
            //
            /*подписываемся на события combobox для рисования изображений*/
            this->imgToolStripComboBox->DropDownWidth = this->imgToolStripComboBox-
>Size.Width;
            cbItemWidth = this->imgToolStripComboBox->DropDownWidth;
            cbItemHeight = (cbItemWidth*3)>>2;
            this->imgToolStripComboBox->DropDownHeight = 1000;
            this->imgToolStripComboBox->ComboBox->DrawMode                          =
System::Windows::Forms::DrawMode::OwnerDrawVariable;
            //this->imgToolStripComboBox->ComboBox->ItemHeight = 50;
            this->imgToolStripComboBox->ComboBox->DrawItem          +=          gcnew
DrawItemEventHandler(this,&MainForm::imgToolStripComboBox_DrawItem);
            this->imgToolStripComboBox->ComboBox->MeasureItem        +=        gcnew
MeasureItemEventHandler(this,&MainForm::imgToolStripComboBox_MeasureItem);
            /*подписываемся на событие прокрутки мыши*/
            this->MouseWheel  +=  gcnew  System::Windows::Forms::MouseEventHandler(
this, &MainForm::MainForm_MouseWheel);
            /*создаем настрки*/
            mainFormSettings = gcnew MainFormSettings();
            /*инструменты (инициализируем руку)*/
            tEditToolStripItem_Click(nullptr,nullptr);
            /*навигатор*/
            navigatorForm = gcnew NavigatorForm();
            // установим владельца
            navigatorForm->Owner = this;
            //подписсываемся на событие
            navigatorForm->NavigatorRectangleMove              +=              gcnew
NavigatorRectangleEventHandler(this,&MainForm::navigatorForm_NavigatorRectangleMove);
            navigatorForm->NeedGraphicsData                  +=                  gcnew
EventHandler(this,&MainForm::navigatorForm_NeedGraphicsData);
            navigatorForm->VisibleChanged                    +=                  gcnew
System::EventHandler(this,&MainForm::navigatorForm_VisibleChanged);
            /*настройки*/
            settingsForm = gcnew SettingsForm();
            // установим владельца
            settingsForm->Owner = this;
            // подписываемся на событие
            settingsForm->VisibleChanged                   +=                   gcnew
System::EventHandler(this,&MainForm::settingsForm_VisibleChanged);
            /*отчет*/
            reportForm = gcnew ReportForm();
            /*3d plot*/
            plot3DForm = gcnew Plot3DForm();
            // установим владельца
            plot3DForm->Owner = this;
            // подписываемся на событие
            plot3DForm->VisibleChanged                  +=                   gcnew
System::EventHandler(this,&MainForm::plot3DForm_VisibleChanged);
            // Устанавливаем режим обновления окна
            this->SetStyle(ControlStyles::UserPaint,true);
            this->SetStyle(ControlStyles::AllPaintingInWmPaint,true);
            this->SetStyle(ControlStyles::DoubleBuffer,true);
            this->SetStyle(ControlStyles::OptimizedDoubleBuffer,true);
            /*создание делегатов для асинхронного выполнения действий на форме*/
            setBusyStyle                              =                           gcnew
SetBusyStyleDelegate(this,&MainForm::setBusyStyleMethod);
            resetBusyStyle                             =                          gcnew
ResetBusyStyleDelegate(this,&MainForm::resetBusyStyleMethod);
            /*присобачить выбор инструмета на контекстное меню */
      }
```

```
protected:
	/// <summary>
	/// Освободить все используемые ресурсы.
	/// </summary>
	~MainForm()
	{
		if (components)
		{
			delete components;
		}
	}
#pragma region дочерние окна
private:
	NavigatorForm^ navigatorForm;
	SettingsForm^ settingsForm;
	ReportForm^ reportForm;
	Plot3DForm^ plot3DForm;
#pragma endregion дочерние окна
#pragma region поля
private:
	Image^ mainImage;
	System::Drawing::Size minSizeImage;
	System::Drawing::Size maxSizeImage;
	Single deltaZoom;
	BaseToolState^ currentToolState;
	MainPresenter^ presenter;
	MainFormSettings^ mainFormSettings;
	delegate System::Void SetBusyStyleDelegate(String^ info);
	SetBusyStyleDelegate^ setBusyStyle;
	delegate System::Void ResetBusyStyleDelegate();
	ResetBusyStyleDelegate^ resetBusyStyle;
	IResultData^ graphicsData;
	/*static*/ initonly Int32 cbItemHeight; // высота картинки в combobox
	/*static*/ initonly Int32 cbItemWidth; // ширина картинки в combobox
#pragma endregion поля
private: System::Windows::Forms::MenuStrip^  menuStrip;
protected:

private: System::Windows::Forms::ToolStripMenuItem^  fileToolStripMenuItem;
protected:
private: System::Windows::Forms::ToolStripMenuItem^  openToolStripMenuItem;
private: System::Windows::Forms::ToolStripSeparator^  toolStripSeparator;
private: System::Windows::Forms::ToolStripMenuItem^  exitToolStripMenuItem;


private: System::Windows::Forms::ToolStripMenuItem^  serviceToolStripMenuItem;
private: System::Windows::Forms::ToolStripMenuItem^  settingsToolStripMenuItem;
private: System::Windows::Forms::ToolStripMenuItem^  navigatorToolStripMenuItem;

private: System::Windows::Forms::ToolStripMenuItem^  helpToolStripMenuItem;
private: System::Windows::Forms::ToolStripMenuItem^  aboutToolStripMenuItem;
private: System::Windows::Forms::ToolStrip^  toolStrip;
private: System::Windows::Forms::ToolStripButton^  reportToolStripButton;

private: System::Windows::Forms::ToolStripButton^  zoom_inToolStripButton;
private: System::Windows::Forms::ToolStripSeparator^  toolStripSeparator6;
private: System::Windows::Forms::ToolStripButton^  zoom_outToolStripButton;
private: System::Windows::Forms::ToolStripButton^  zoom_fitToolStripButton;
private: System::Windows::Forms::ToolStripButton^  zoom_100ToolStripButton;
```

```
private: System::Windows::Forms::StatusStrip^  statusStrip;
private: System::Windows::Forms::Panel^  workspacePanel;

private: System::Windows::Forms::PictureBox^  mainCanvas;
private: System::Windows::Forms::ToolStripMenuItem^  reportToolStripMenuItem;
private: System::Windows::Forms::ToolStripMenuItem^  plot3DToolStripMenuItem;
private: System::Windows::Forms::ToolStripStatusLabel^  toolsToolStripStatusLabel;
private: System::Windows::Forms::ToolStripProgressBar^  toolStripProgressBar;




private: System::Windows::Forms::ToolStripSeparator^  toolStripSeparator1;
private: System::Windows::Forms::ToolStripButton^  runToolStripButton;
private: System::Windows::Forms::ToolTip^  toolTip;
private: System::Windows::Forms::Timer^  timerForToolTip;
private: System::Windows::Forms::ToolStripStatusLabel^  infoToolStripStatusLabel;
private: System::Windows::Forms::ToolStripStatusLabel^  zoomToolStripStatusLabel;
private: System::Windows::Forms::ContextMenuStrip^  contextMenuStrip;
private: System::Windows::Forms::ToolStripStatusLabel^  resultToolStripStatusLabel;
private: System::Windows::Forms::ToolStripComboBox^  imgToolStripComboBox;
private: System::Windows::Forms::ToolStripMenuItem^  contentToolStripMenuItem;
private: System::Windows::Forms::ToolStripButton^  openToolStripButton;
private: System::Windows::Forms::ToolStripButton^  helpToolStripButton;
private: System::Windows::Forms::ToolStripButton^  editToolStripButton;
private: System::Windows::Forms::ToolStripButton^  calibrateToolStripButton;
private: System::Windows::Forms::ToolStripSeparator^  toolStripSeparator2;
private: System::Windows::Forms::ToolStripSeparator^  toolStripSeparator3;
private: System::Windows::Forms::ToolStripMenuItem^  панельToolStripMenuItem;
private: System::Windows::Forms::ToolStripMenuItem^  z_inToolStripMenuItem;
private: System::Windows::Forms::ToolStripMenuItem^  z_outToolStripMenuItem;
private: System::Windows::Forms::ToolStripMenuItem^  z_fitToolStripMenuItem;
private: System::Windows::Forms::ToolStripMenuItem^  z_normToolStripMenuItem;
private: System::Windows::Forms::ToolStripSeparator^  toolStripMenuItem1;
private: System::Windows::Forms::ToolStripMenuItem^  запускToolStripMenuItem;
private: System::Windows::Forms::ToolStripMenuItem^  редакторToolStripMenuItem;
private: System::Windows::Forms::ToolStripMenuItem^  калибровкаToolStripMenuItem;
private: System::Diagnostics::Process^  helpProcess;
private: System::Windows::Forms::ToolStripSeparator^  toolStripMenuItem2;


private: System::ComponentModel::IContainer^  components;
private:
	/// <summary>
	/// Требуется переменная конструктора.
	/// </summary>

#pragma region Windows Form Designer generated code
	/// <summary>
	/// Обязательный метод для поддержки конструктора - не изменяйте
	/// содержимое данного метода при помощи редактора кода.
	/// </summary>
	void InitializeComponent(void)
	{
		this->components = (gcnew System::ComponentModel::Container());
		System::ComponentModel::ComponentResourceManager^   resources  =  (gcnew
System::ComponentModel::ComponentResourceManager(MainForm::typeid));
		this->menuStrip = (gcnew System::Windows::Forms::MenuStrip());
		this->contextMenuStrip                    =                    (gcnew
System::Windows::Forms::ContextMenuStrip(this->components));
```

```
            this->fileToolStripMenuItem                  =                  (gcnew
System::Windows::Forms::ToolStripMenuItem());
            this->openToolStripMenuItem                  =                  (gcnew
System::Windows::Forms::ToolStripMenuItem());
            this->reportToolStripMenuItem                =                  (gcnew
System::Windows::Forms::ToolStripMenuItem());
            this->toolStripSeparator                    =                  (gcnew
System::Windows::Forms::ToolStripSeparator());
            this->exitToolStripMenuItem                  =                  (gcnew
System::Windows::Forms::ToolStripMenuItem());
            this->serviceToolStripMenuItem               =                  (gcnew
System::Windows::Forms::ToolStripMenuItem());
            this->navigatorToolStripMenuItem             =                  (gcnew
System::Windows::Forms::ToolStripMenuItem());
            this->plot3DToolStripMenuItem                =                  (gcnew
System::Windows::Forms::ToolStripMenuItem());
            this->toolStripMenuItem2                    =                  (gcnew
System::Windows::Forms::ToolStripSeparator());
            this->settingsToolStripMenuItem              =                  (gcnew
System::Windows::Forms::ToolStripMenuItem());
            this->helpToolStripMenuItem                  =                  (gcnew
System::Windows::Forms::ToolStripMenuItem());
            this->contentToolStripMenuItem               =                  (gcnew
System::Windows::Forms::ToolStripMenuItem());
            this->aboutToolStripMenuItem                 =                  (gcnew
System::Windows::Forms::ToolStripMenuItem());
            this->панельToolStripMenuItem                =                  (gcnew
System::Windows::Forms::ToolStripMenuItem());
            this->z_inToolStripMenuItem                  =                  (gcnew
System::Windows::Forms::ToolStripMenuItem());
            this->z_outToolStripMenuItem                 =                  (gcnew
System::Windows::Forms::ToolStripMenuItem());
            this->z_fitToolStripMenuItem                 =                  (gcnew
System::Windows::Forms::ToolStripMenuItem());
            this->z_normToolStripMenuItem                =                  (gcnew
System::Windows::Forms::ToolStripMenuItem());
            this->toolStripMenuItem1                    =                  (gcnew
System::Windows::Forms::ToolStripSeparator());
            this->запускToolStripMenuItem                =                  (gcnew
System::Windows::Forms::ToolStripMenuItem());
            this->редакторToolStripMenuItem              =                  (gcnew
System::Windows::Forms::ToolStripMenuItem());
            this->калибровкаToolStripMenuItem            =                  (gcnew
System::Windows::Forms::ToolStripMenuItem());
            this->toolStrip = (gcnew System::Windows::Forms::ToolStrip());
            this->openToolStripButton                   =                  (gcnew
System::Windows::Forms::ToolStripButton());
            this->reportToolStripButton                  =                  (gcnew
System::Windows::Forms::ToolStripButton());
            this->toolStripSeparator6                   =                  (gcnew
System::Windows::Forms::ToolStripSeparator());
            this->runToolStripButton                    =                  (gcnew
System::Windows::Forms::ToolStripButton());
            this->editToolStripButton                   =                  (gcnew
System::Windows::Forms::ToolStripButton());
            this->calibrateToolStripButton               =                  (gcnew
System::Windows::Forms::ToolStripButton());
            this->toolStripSeparator1                   =                  (gcnew
System::Windows::Forms::ToolStripSeparator());
            this->imgToolStripComboBox                   =                  (gcnew
System::Windows::Forms::ToolStripComboBox());
```

```
            this->toolStripSeparator3                    =                    (gcnew
System::Windows::Forms::ToolStripSeparator());
            this->zoom_inToolStripButton                   =                    (gcnew
System::Windows::Forms::ToolStripButton());
            this->zoom_outToolStripButton                  =                    (gcnew
System::Windows::Forms::ToolStripButton());
            this->zoom_fitToolStripButton                  =                    (gcnew
System::Windows::Forms::ToolStripButton());
            this->zoom_100ToolStripButton                  =                    (gcnew
System::Windows::Forms::ToolStripButton());
            this->toolStripSeparator2                    =                    (gcnew
System::Windows::Forms::ToolStripSeparator());
            this->helpToolStripButton                    =                    (gcnew
System::Windows::Forms::ToolStripButton());
            this->statusStrip = (gcnew System::Windows::Forms::StatusStrip());
            this->toolStripProgressBar                    =                    (gcnew
System::Windows::Forms::ToolStripProgressBar());
            this->infoToolStripStatusLabel                 =                    (gcnew
System::Windows::Forms::ToolStripStatusLabel());
            this->resultToolStripStatusLabel                =                    (gcnew
System::Windows::Forms::ToolStripStatusLabel());
            this->toolsToolStripStatusLabel                =                    (gcnew
System::Windows::Forms::ToolStripStatusLabel());
            this->zoomToolStripStatusLabel                 =                    (gcnew
System::Windows::Forms::ToolStripStatusLabel());
            this->workspacePanel = (gcnew System::Windows::Forms::Panel());
            this->mainCanvas = (gcnew System::Windows::Forms::PictureBox());
            this->toolTip     =     (gcnew     System::Windows::Forms::ToolTip(this-
>components));
            this->timerForToolTip   =   (gcnew   System::Windows::Forms::Timer(this-
>components));
            this->helpProcess = (gcnew System::Diagnostics::Process());
            this->menuStrip->SuspendLayout();
            this->toolStrip->SuspendLayout();
            this->statusStrip->SuspendLayout();
            this->workspacePanel->SuspendLayout();
            (cli::safe_cast<System::ComponentModel::ISupportInitialize^      >(this-
>mainCanvas))->BeginInit();
            this->SuspendLayout();
            //
            // menuStrip
            //
            this->menuStrip->ContextMenuStrip = this->contextMenuStrip;
            this->menuStrip->Items->AddRange(gcnew                        cli::array<
System::Windows::Forms::ToolStripItem^  >(4) {this->fileToolStripMenuItem,
                this->serviceToolStripMenuItem,      this->helpToolStripMenuItem,
this->панельToolStripMenuItem});
            this->menuStrip->Location = System::Drawing::Point(0, 0);
            this->menuStrip->Name = L"menuStrip";
            this->menuStrip->RenderMode                                                =
System::Windows::Forms::ToolStripRenderMode::Professional;
            this->menuStrip->Size = System::Drawing::Size(784, 24);
            this->menuStrip->TabIndex = 0;
            this->menuStrip->Text = L"menuStrip1";
            //
            // contextMenuStrip
            //
            this->contextMenuStrip->Name = L"contextMenuStrip";
            this->contextMenuStrip->Size = System::Drawing::Size(61, 4);
            //
            // fileToolStripMenuItem
```

```
            //
            this->fileToolStripMenuItem->DropDownItems->AddRange(gcnew   cli::array<
System::Windows::Forms::ToolStripItem^  >(4) {this->openToolStripMenuItem,
                  this->reportToolStripMenuItem,   this->toolStripSeparator,   this-
>exitToolStripMenuItem});
            this->fileToolStripMenuItem->Name = L"fileToolStripMenuItem";
            this->fileToolStripMenuItem->Size = System::Drawing::Size(48, 20);
            this->fileToolStripMenuItem->Text = L"&Файл";
            //
            // openToolStripMenuItem
            //
            this->openToolStripMenuItem->ImageTransparentColor                    =
System::Drawing::Color::Magenta;
            this->openToolStripMenuItem->Name = L"openToolStripMenuItem";
            this->openToolStripMenuItem->ShortcutKeys                             =
static_cast<System::Windows::Forms::Keys>((System::Windows::Forms::Keys::Control    |
System::Windows::Forms::Keys::O));
            this->openToolStripMenuItem->Size = System::Drawing::Size(164, 22);
            this->openToolStripMenuItem->Text = L"&Открыть";
            this->openToolStripMenuItem->Click  +=  gcnew  System::EventHandler(this,
&MainForm::openToolStripMenuItem_Click);
            //
            // reportToolStripMenuItem
            //
            this->reportToolStripMenuItem->Name = L"reportToolStripMenuItem";
            this->reportToolStripMenuItem->ShortcutKeys                           =
static_cast<System::Windows::Forms::Keys>((System::Windows::Forms::Keys::Control    |
System::Windows::Forms::Keys::R));
            this->reportToolStripMenuItem->Size = System::Drawing::Size(164, 22);
            this->reportToolStripMenuItem->Text = L"&Отчет";
            this->reportToolStripMenuItem->Click += gcnew System::EventHandler(this,
&MainForm::reportToolStripMenuItem_Click);
            //
            // toolStripSeparator
            //
            this->toolStripSeparator->Name = L"toolStripSeparator";
            this->toolStripSeparator->Size = System::Drawing::Size(161, 6);
            //
            // exitToolStripMenuItem
            //
            this->exitToolStripMenuItem->Name = L"exitToolStripMenuItem";
            this->exitToolStripMenuItem->Size = System::Drawing::Size(164, 22);
            this->exitToolStripMenuItem->Text = L"Вы&ход";
            //
            // serviceToolStripMenuItem
            //
            this->serviceToolStripMenuItem->DropDownItems->AddRange(gcnew
cli::array<      System::Windows::Forms::ToolStripItem^             >(4)      {this-
>navigatorToolStripMenuItem,
                  this->plot3DToolStripMenuItem,   this->toolStripMenuItem2,   this-
>settingsToolStripMenuItem});
            this->serviceToolStripMenuItem->Name = L"serviceToolStripMenuItem";
            this->serviceToolStripMenuItem->Size = System::Drawing::Size(59, 20);
            this->serviceToolStripMenuItem->Text = L"&Сервис";
            //
            // navigatorToolStripMenuItem
            //
            this->navigatorToolStripMenuItem->Name = L"navigatorToolStripMenuItem";
            this->navigatorToolStripMenuItem->Size = System::Drawing::Size(134, 22);
            this->navigatorToolStripMenuItem->Text = L"&Навигатор";
```

```
            this->navigatorToolStripMenuItem->Click            +=            gcnew
System::EventHandler(this, &MainForm::navigatorToolStripMenuItem_Click);
            //
            // plot3DToolStripMenuItem
            //
            this->plot3DToolStripMenuItem->Name = L"plot3DToolStripMenuItem";
            this->plot3DToolStripMenuItem->Size = System::Drawing::Size(134, 22);
            this->plot3DToolStripMenuItem->Text = L"3D модель";
            this->plot3DToolStripMenuItem->Click += gcnew System::EventHandler(this,
&MainForm::plot3DToolStripMenuItem_Click);
            //
            // toolStripMenuItem2
            //
            this->toolStripMenuItem2->Name = L"toolStripMenuItem2";
            this->toolStripMenuItem2->Size = System::Drawing::Size(131, 6);
            //
            // settingsToolStripMenuItem
            //
            this->settingsToolStripMenuItem->Name = L"settingsToolStripMenuItem";
            this->settingsToolStripMenuItem->Size = System::Drawing::Size(134, 22);
            this->settingsToolStripMenuItem->Text = L"&Настройки";
            this->settingsToolStripMenuItem->Click             +=            gcnew
System::EventHandler(this, &MainForm::settingsToolStripMenuItem_Click);
            //
            // helpToolStripMenuItem
            //
            this->helpToolStripMenuItem->DropDownItems->AddRange(gcnew   cli::array<
System::Windows::Forms::ToolStripItem^  >(2) {this->contentToolStripMenuItem,
                  this->aboutToolStripMenuItem});
            this->helpToolStripMenuItem->Name = L"helpToolStripMenuItem";
            this->helpToolStripMenuItem->Size = System::Drawing::Size(65, 20);
            this->helpToolStripMenuItem->Text = L"Спра&вка";
            //
            // contentToolStripMenuItem
            //
            this->contentToolStripMenuItem->Name = L"contentToolStripMenuItem";
            this->contentToolStripMenuItem->ShortcutKeys                              =
static_cast<System::Windows::Forms::Keys>((System::Windows::Forms::Keys::Control    |
System::Windows::Forms::Keys::F1));
            this->contentToolStripMenuItem->Size = System::Drawing::Size(189, 22);
            this->contentToolStripMenuItem->Text = L"&Содержание";
            this->contentToolStripMenuItem->Click             +=            gcnew
System::EventHandler(this, &MainForm::helpToolStripButton_Click);
            //
            // aboutToolStripMenuItem
            //
            this->aboutToolStripMenuItem->Name = L"aboutToolStripMenuItem";
            this->aboutToolStripMenuItem->Size = System::Drawing::Size(189, 22);
            this->aboutToolStripMenuItem->Text = L"&О программе...";
            this->aboutToolStripMenuItem->Click += gcnew System::EventHandler(this,
&MainForm::aboutToolStripMenuItem_Click);
            //
            // панельToolStripMenuItem
            //
            this->панельToolStripMenuItem->DropDownItems->AddRange(gcnew cli::array<
System::Windows::Forms::ToolStripItem^  >(8) {this->z_inToolStripMenuItem,
                  this->z_outToolStripMenuItem, this->z_fitToolStripMenuItem, this-
>z_normToolStripMenuItem, this->toolStripMenuItem1, this->запускToolStripMenuItem,
                  this->редакторToolStripMenuItem,                              this-
>калибровкаToolStripMenuItem});
            this->панельToolStripMenuItem->Name = L"панельToolStripMenuItem";
```

```
            this->панельToolStripMenuItem->Size = System::Drawing::Size(60, 20);
            this->панельToolStripMenuItem->Text = L"Панель";
            this->панельToolStripMenuItem->Visible = false;
            //
            // z_inToolStripMenuItem
            //
            this->z_inToolStripMenuItem->Name = L"z_inToolStripMenuItem";
            this->z_inToolStripMenuItem->ShortcutKeys                                    =
static_cast<System::Windows::Forms::Keys>((System::Windows::Forms::Keys::Control    |
System::Windows::Forms::Keys::Oemplus));
            this->z_inToolStripMenuItem->Size = System::Drawing::Size(269, 22);
            this->z_inToolStripMenuItem->Text = L"Увеличить масш.";
            this->z_inToolStripMenuItem->Click  +=  gcnew  System::EventHandler(this,
&MainForm::zoom_inToolStripButton_Click);
            //
            // z_outToolStripMenuItem
            //
            this->z_outToolStripMenuItem->Name = L"z_outToolStripMenuItem";
            this->z_outToolStripMenuItem->ShortcutKeys                                   =
static_cast<System::Windows::Forms::Keys>((System::Windows::Forms::Keys::Control    |
System::Windows::Forms::Keys::OemMinus));
            this->z_outToolStripMenuItem->Size = System::Drawing::Size(269, 22);
            this->z_outToolStripMenuItem->Text = L"Уменьшить масш.";
            this->z_outToolStripMenuItem->Click  +=  gcnew  System::EventHandler(this,
&MainForm::zoom_outToolStripButton_Click);
            //
            // z_fitToolStripMenuItem
            //
            this->z_fitToolStripMenuItem->Name = L"z_fitToolStripMenuItem";
            this->z_fitToolStripMenuItem->ShortcutKeys                                   =
static_cast<System::Windows::Forms::Keys>((System::Windows::Forms::Keys::Control    |
System::Windows::Forms::Keys::F));
            this->z_fitToolStripMenuItem->Size = System::Drawing::Size(269, 22);
            this->z_fitToolStripMenuItem->Text = L"Развернуть на все";
            this->z_fitToolStripMenuItem->Click  +=  gcnew  System::EventHandler(this,
&MainForm::zoom_fitToolStripButton_Click);
            //
            // z_normToolStripMenuItem
            //
            this->z_normToolStripMenuItem->Name = L"z_normToolStripMenuItem";
            this->z_normToolStripMenuItem->ShortcutKeys                                  =
static_cast<System::Windows::Forms::Keys>((System::Windows::Forms::Keys::Control    |
System::Windows::Forms::Keys::N));
            this->z_normToolStripMenuItem->Size = System::Drawing::Size(269, 22);
            this->z_normToolStripMenuItem->Text = L"Масштаб 100";
            this->z_normToolStripMenuItem->Click += gcnew System::EventHandler(this,
&MainForm::zoom_100ToolStripButton_Click);
            //
            // toolStripMenuItem1
            //
            this->toolStripMenuItem1->Name = L"toolStripMenuItem1";
            this->toolStripMenuItem1->Size = System::Drawing::Size(266, 6);
            //
            // запускToolStripMenuItem
            //
            this->запускToolStripMenuItem->Name = L"запускToolStripMenuItem";
            this->запускToolStripMenuItem->ShortcutKeys                                  =
static_cast<System::Windows::Forms::Keys>((System::Windows::Forms::Keys::Control    |
System::Windows::Forms::Keys::F5));
            this->запускToolStripMenuItem->Size = System::Drawing::Size(269, 22);
            this->запускToolStripMenuItem->Text = L"Запуск";
```

```
            this->запускToolStripMenuItem->Click += gcnew System::EventHandler(this,
&MainForm::runToolStripButton_Click);
            //
            // редакторToolStripMenuItem
            //
            this->редакторToolStripMenuItem->Name = L"редакторToolStripMenuItem";
            this->редакторToolStripMenuItem->Size = System::Drawing::Size(269, 22);
            this->редакторToolStripMenuItem->Text = L"Редактор";
            //
            // калибровкаToolStripMenuItem
            //
            this->калибровкаToolStripMenuItem->Name                                =
L"калибровкаToolStripMenuItem";
            this->калибровкаToolStripMenuItem->Size    =    System::Drawing::Size(269,
22);
            this->калибровкаToolStripMenuItem->Text = L"Калибровка";
            //
            // toolStrip
            //
            this->toolStrip->AutoSize = false;
            this->toolStrip->ImageScalingSize = System::Drawing::Size(32, 32);
            this->toolStrip->Items->AddRange(gcnew                          cli::array<
System::Windows::Forms::ToolStripItem^  >(15) {this->openToolStripButton,
                this->reportToolStripButton,    this->toolStripSeparator6,    this-
>runToolStripButton, this->editToolStripButton, this->calibrateToolStripButton,
                this->toolStripSeparator1,    this->imgToolStripComboBox,    this-
>toolStripSeparator3, this->zoom_inToolStripButton, this->zoom_outToolStripButton,
                this->zoom_fitToolStripButton,      this->zoom_100ToolStripButton,
this->toolStripSeparator2, this->helpToolStripButton});
            this->toolStrip->Location = System::Drawing::Point(0, 24);
            this->toolStrip->Name = L"toolStrip";
            this->toolStrip->RenderMode                                            =
System::Windows::Forms::ToolStripRenderMode::System;
            this->toolStrip->Size = System::Drawing::Size(784, 35);
            this->toolStrip->TabIndex = 1;
            this->toolStrip->Text = L"toolStrip1";
            //
            // openToolStripButton
            //
            this->openToolStripButton->DisplayStyle                                =
System::Windows::Forms::ToolStripItemDisplayStyle::Image;
            this->openToolStripButton->Image                                       =
(cli::safe_cast<System::Drawing::Image^                                >(resources-
>GetObject(L"openToolStripButton.Image")));
            this->openToolStripButton->ImageTransparentColor                       =
System::Drawing::Color::Magenta;
            this->openToolStripButton->Name = L"openToolStripButton";
            this->openToolStripButton->Size = System::Drawing::Size(36, 32);
            this->openToolStripButton->Text = L"&Открыть";
            this->openToolStripButton->Click   +=   gcnew   System::EventHandler(this,
&MainForm::openToolStripMenuItem_Click);
            //
            // reportToolStripButton
            //
            this->reportToolStripButton->DisplayStyle                              =
System::Windows::Forms::ToolStripItemDisplayStyle::Image;
            this->reportToolStripButton->Image                                     =
(cli::safe_cast<System::Drawing::Image^                                >(resources-
>GetObject(L"reportToolStripButton.Image")));
            this->reportToolStripButton->ImageTransparentColor                     =
System::Drawing::Color::Magenta;
```

```
            this->reportToolStripButton->Name = L"reportToolStripButton";
            this->reportToolStripButton->Size = System::Drawing::Size(36, 32);
            this->reportToolStripButton->Text = L"&Отчет";
            this->reportToolStripButton->Click  +=  gcnew  System::EventHandler(this,
&MainForm::reportToolStripMenuItem_Click);
            //
            // toolStripSeparator6
            //
            this->toolStripSeparator6->Name = L"toolStripSeparator6";
            this->toolStripSeparator6->Size = System::Drawing::Size(6, 35);
            //
            // runToolStripButton
            //
            this->runToolStripButton->DisplayStyle                                    =
System::Windows::Forms::ToolStripItemDisplayStyle::Image;
            this->runToolStripButton->Image                                           =
(cli::safe_cast<System::Drawing::Image^                                  >(resources-
>GetObject(L"runToolStripButton.Image")));
            this->runToolStripButton->ImageTransparentColor                           =
System::Drawing::Color::Magenta;
            this->runToolStripButton->Name = L"runToolStripButton";
            this->runToolStripButton->Size = System::Drawing::Size(36, 32);
            this->runToolStripButton->Text = L"Запуск";
            this->runToolStripButton->Click   +=   gcnew   System::EventHandler(this,
&MainForm::runToolStripButton_Click);
            //
            // editToolStripButton
            //
            this->editToolStripButton->DisplayStyle                                   =
System::Windows::Forms::ToolStripItemDisplayStyle::Image;
            this->editToolStripButton->Image                                          =
(cli::safe_cast<System::Drawing::Image^                                  >(resources-
>GetObject(L"editToolStripButton.Image")));
            this->editToolStripButton->ImageTransparentColor                          =
System::Drawing::Color::Magenta;
            this->editToolStripButton->Name = L"editToolStripButton";
            this->editToolStripButton->Size = System::Drawing::Size(36, 32);
            this->editToolStripButton->Text = L"Редактор";
            this->editToolStripButton->Click   +=   gcnew   System::EventHandler(this,
&MainForm::tEditToolStripItem_Click);
            //
            // calibrateToolStripButton
            //
            this->calibrateToolStripButton->DisplayStyle                              =
System::Windows::Forms::ToolStripItemDisplayStyle::Image;
            this->calibrateToolStripButton->Image                                     =
(cli::safe_cast<System::Drawing::Image^                                  >(resources-
>GetObject(L"calibrateToolStripButton.Image")));
            this->calibrateToolStripButton->ImageTransparentColor                     =
System::Drawing::Color::Magenta;
            this->calibrateToolStripButton->Name = L"calibrateToolStripButton";
            this->calibrateToolStripButton->Size = System::Drawing::Size(36, 32);
            this->calibrateToolStripButton->Text = L"Калибровка";
            this->calibrateToolStripButton->Click                 +=                gcnew
System::EventHandler(this, &MainForm::tCalibrateToolStripItem_Click);
            //
            // toolStripSeparator1
            //
            this->toolStripSeparator1->Name = L"toolStripSeparator1";
            this->toolStripSeparator1->Size = System::Drawing::Size(6, 35);
            //
```

```
            // imgToolStripComboBox
            //
            this->imgToolStripComboBox->BackColor                                    =
System::Drawing::SystemColors::Window;
            this->imgToolStripComboBox->DropDownWidth = 200;
            this->imgToolStripComboBox->FlatStyle                                    =
System::Windows::Forms::FlatStyle::Flat;
            this->imgToolStripComboBox->Font = (gcnew System::Drawing::Font(L"Segoe
UI", 12));
            this->imgToolStripComboBox->IntegralHeight = false;
            this->imgToolStripComboBox->Name = L"imgToolStripComboBox";
            this->imgToolStripComboBox->Size = System::Drawing::Size(165, 35);
            this->imgToolStripComboBox->SelectedIndexChanged        +=        gcnew
System::EventHandler(this, &MainForm::imgToolStripComboBox_SelectedIndexChanged);
            //
            // toolStripSeparator3
            //
            this->toolStripSeparator3->Name = L"toolStripSeparator3";
            this->toolStripSeparator3->Size = System::Drawing::Size(6, 35);
            //
            // zoom_inToolStripButton
            //
            this->zoom_inToolStripButton->DisplayStyle                               =
System::Windows::Forms::ToolStripItemDisplayStyle::Image;
            this->zoom_inToolStripButton->Image                                      =
(cli::safe_cast<System::Drawing::Image^                                  >(resources-
>GetObject(L"zoom_inToolStripButton.Image")));
            this->zoom_inToolStripButton->ImageTransparentColor                      =
System::Drawing::Color::Magenta;
            this->zoom_inToolStripButton->Name = L"zoom_inToolStripButton";
            this->zoom_inToolStripButton->Size = System::Drawing::Size(36, 32);
            this->zoom_inToolStripButton->Text = L"Увеличить масштаб";
            this->zoom_inToolStripButton->Click += gcnew System::EventHandler(this,
&MainForm::zoom_inToolStripButton_Click);
            //
            // zoom_outToolStripButton
            //
            this->zoom_outToolStripButton->DisplayStyle                              =
System::Windows::Forms::ToolStripItemDisplayStyle::Image;
            this->zoom_outToolStripButton->Image                                     =
(cli::safe_cast<System::Drawing::Image^                                  >(resources-
>GetObject(L"zoom_outToolStripButton.Image")));
            this->zoom_outToolStripButton->ImageTransparentColor                     =
System::Drawing::Color::Magenta;
            this->zoom_outToolStripButton->Name = L"zoom_outToolStripButton";
            this->zoom_outToolStripButton->Size = System::Drawing::Size(36, 32);
            this->zoom_outToolStripButton->Text = L"Уменьшить масштаб";
            this->zoom_outToolStripButton->Click += gcnew System::EventHandler(this,
&MainForm::zoom_outToolStripButton_Click);
            //
            // zoom_fitToolStripButton
            //
            this->zoom_fitToolStripButton->DisplayStyle                              =
System::Windows::Forms::ToolStripItemDisplayStyle::Image;
            this->zoom_fitToolStripButton->Image                                     =
(cli::safe_cast<System::Drawing::Image^                                  >(resources-
>GetObject(L"zoom_fitToolStripButton.Image")));
            this->zoom_fitToolStripButton->ImageTransparentColor                     =
System::Drawing::Color::Magenta;
            this->zoom_fitToolStripButton->Name = L"zoom_fitToolStripButton";
            this->zoom_fitToolStripButton->Size = System::Drawing::Size(36, 32);
```

```
            this->zoom_fitToolStripButton->Text = L"Развернуть на все окно";
            this->zoom_fitToolStripButton->Click += gcnew System::EventHandler(this,
&MainForm::zoom_fitToolStripButton_Click);
            //
            // zoom_100ToolStripButton
            //
            this->zoom_100ToolStripButton->DisplayStyle                          =
System::Windows::Forms::ToolStripItemDisplayStyle::Image;
            this->zoom_100ToolStripButton->Image                                 =
(cli::safe_cast<System::Drawing::Image^                                >(resources-
>GetObject(L"zoom_100ToolStripButton.Image")));
            this->zoom_100ToolStripButton->ImageTransparentColor                 =
System::Drawing::Color::Magenta;
            this->zoom_100ToolStripButton->Name = L"zoom_100ToolStripButton";
            this->zoom_100ToolStripButton->Size = System::Drawing::Size(36, 32);
            this->zoom_100ToolStripButton->Text = L"Масштаб 100%";
            this->zoom_100ToolStripButton->Click += gcnew System::EventHandler(this,
&MainForm::zoom_100ToolStripButton_Click);
            //
            // toolStripSeparator2
            //
            this->toolStripSeparator2->Name = L"toolStripSeparator2";
            this->toolStripSeparator2->Size = System::Drawing::Size(6, 35);
            //
            // helpToolStripButton
            //
            this->helpToolStripButton->DisplayStyle                              =
System::Windows::Forms::ToolStripItemDisplayStyle::Image;
            this->helpToolStripButton->Image                                     =
(cli::safe_cast<System::Drawing::Image^                                >(resources-
>GetObject(L"helpToolStripButton.Image")));
            this->helpToolStripButton->ImageTransparentColor                     =
System::Drawing::Color::Magenta;
            this->helpToolStripButton->Name = L"helpToolStripButton";
            this->helpToolStripButton->Size = System::Drawing::Size(36, 32);
            this->helpToolStripButton->Text = L"Спр&авка";
            this->helpToolStripButton->Click   +=   gcnew   System::EventHandler(this,
&MainForm::helpToolStripButton_Click);
            //
            // statusStrip
            //
            this->statusStrip->AutoSize = false;
            this->statusStrip->ImageScalingSize = System::Drawing::Size(32, 32);
            this->statusStrip->Items->AddRange(gcnew                      cli::array<
System::Windows::Forms::ToolStripItem^  >(5) {this->toolStripProgressBar,
                this->infoToolStripStatusLabel,  this->resultToolStripStatusLabel,
this->toolsToolStripStatusLabel, this->zoomToolStripStatusLabel});
            this->statusStrip->Location = System::Drawing::Point(0, 527);
            this->statusStrip->Name = L"statusStrip";
            this->statusStrip->RenderMode                                        =
System::Windows::Forms::ToolStripRenderMode::Professional;
            this->statusStrip->ShowItemToolTips = true;
            this->statusStrip->Size = System::Drawing::Size(784, 35);
            this->statusStrip->TabIndex = 2;
            this->statusStrip->Text = L"statusStrip1";
            //
            // toolStripProgressBar
            //
            this->toolStripProgressBar->Name = L"toolStripProgressBar";
            this->toolStripProgressBar->Size = System::Drawing::Size(200, 29);
            //
```

```
            // infoToolStripStatusLabel
            //
            this->infoToolStripStatusLabel->BorderSides                                 =
System::Windows::Forms::ToolStripStatusLabelBorderSides::Right;
            this->infoToolStripStatusLabel->Font                =                (gcnew
System::Drawing::Font(L"Segoe UI", 12));
            this->infoToolStripStatusLabel->Name = L"infoToolStripStatusLabel";
            this->infoToolStripStatusLabel->Size = System::Drawing::Size(4, 30);
            this->infoToolStripStatusLabel->TextAlign                                   =
System::Drawing::ContentAlignment::MiddleLeft;
            //
            // resultToolStripStatusLabel
            //
            this->resultToolStripStatusLabel->AutoToolTip = true;
            this->resultToolStripStatusLabel->BorderSides                               =
System::Windows::Forms::ToolStripStatusLabelBorderSides::Right;
            this->resultToolStripStatusLabel->Font              =              (gcnew
System::Drawing::Font(L"Segoe UI", 12));
            this->resultToolStripStatusLabel->Name = L"resultToolStripStatusLabel";
            this->resultToolStripStatusLabel->Size = System::Drawing::Size(373, 30);
            this->resultToolStripStatusLabel->Spring = true;
            this->resultToolStripStatusLabel->TextAlign                                 =
System::Drawing::ContentAlignment::MiddleLeft;
            //
            // toolsToolStripStatusLabel
            //
            this->toolsToolStripStatusLabel->Font               =              (gcnew
System::Drawing::Font(L"Segoe UI", 12));
            this->toolsToolStripStatusLabel->Margin                                     =
System::Windows::Forms::Padding(3, 3, 0, 2);
            this->toolsToolStripStatusLabel->Name = L"toolsToolStripStatusLabel";
            this->toolsToolStripStatusLabel->Size = System::Drawing::Size(97, 30);
            this->toolsToolStripStatusLabel->Text = L"Инструмент";
            this->toolsToolStripStatusLabel->ToolTipText = L"Инструмент";
            //
            // zoomToolStripStatusLabel
            //
            this->zoomToolStripStatusLabel->AutoSize = false;
            this->zoomToolStripStatusLabel->BorderSides                                 =
System::Windows::Forms::ToolStripStatusLabelBorderSides::Left;
            this->zoomToolStripStatusLabel->Font                =              (gcnew
System::Drawing::Font(L"Segoe UI", 12));
            this->zoomToolStripStatusLabel->Name = L"zoomToolStripStatusLabel";
            this->zoomToolStripStatusLabel->Size = System::Drawing::Size(90, 30);
            this->zoomToolStripStatusLabel->ToolTipText = L"Масштаб";
            //
            // workspacePanel
            //
            this->workspacePanel->BackColor                                             =
System::Drawing::SystemColors::AppWorkspace;
            this->workspacePanel->Controls->Add(this->mainCanvas);
            this->workspacePanel->Dock = System::Windows::Forms::DockStyle::Fill;
            this->workspacePanel->Location = System::Drawing::Point(0, 59);
            this->workspacePanel->Name = L"workspacePanel";
            this->workspacePanel->Size = System::Drawing::Size(784, 468);
            this->workspacePanel->TabIndex = 8;
            this->workspacePanel->Scroll                    +=                  gcnew
System::Windows::Forms::ScrollEventHandler(this, &MainForm::workspacePanel_Scroll);
            //
            // mainCanvas
            //
```

```
            this->mainCanvas->ContextMenuStrip = this->contextMenuStrip;
            this->mainCanvas->Location = System::Drawing::Point(0, 0);
            this->mainCanvas->Name = L"mainCanvas";
            this->mainCanvas->Size = System::Drawing::Size(10, 10);
            this->mainCanvas->TabIndex = 0;
            this->mainCanvas->TabStop = false;
            this->mainCanvas->Paint                    +=                    gcnew
System::Windows::Forms::PaintEventHandler(this, &MainForm::mainCanvas_Paint);
            this->mainCanvas->MouseClick                +=                    gcnew
System::Windows::Forms::MouseEventHandler(this, &MainForm::mainCanvas_MouseClick);
            this->mainCanvas->MouseDown                 +=                    gcnew
System::Windows::Forms::MouseEventHandler(this, &MainForm::mainCanvas_MouseDown);
            this->mainCanvas->MouseMove                 +=                    gcnew
System::Windows::Forms::MouseEventHandler(this, &MainForm::mainCanvas_MouseMove);
            this->mainCanvas->MouseUp                   +=                    gcnew
System::Windows::Forms::MouseEventHandler(this, &MainForm::mainCanvas_MouseUp);
            //
            // toolTip
            //
            this->toolTip->AutomaticDelay = 300;
            //
            // timerForToolTip
            //
            this->timerForToolTip->Interval = 300;
            this->timerForToolTip->Tick    +=    gcnew    System::EventHandler(this,
&MainForm::timerForToolTip_Tick);
            //
            // helpProcess
            //
            this->helpProcess->StartInfo->Domain = L"";
            this->helpProcess->StartInfo->FileName = L"ForestHelp.exe";
            this->helpProcess->StartInfo->LoadUserProfile = false;
            this->helpProcess->StartInfo->Password = nullptr;
            this->helpProcess->StartInfo->StandardErrorEncoding = nullptr;
            this->helpProcess->StartInfo->StandardOutputEncoding = nullptr;
            this->helpProcess->StartInfo->UserName = L"";
            this->helpProcess->SynchronizingObject = this;
            //
            // MainForm
            //
            this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);
            this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;
            this->ClientSize = System::Drawing::Size(784, 562);
            this->Controls->Add(this->workspacePanel);
            this->Controls->Add(this->statusStrip);
            this->Controls->Add(this->toolStrip);
            this->Controls->Add(this->menuStrip);
            this->DoubleBuffered = true;
            this->Icon   =   (cli::safe_cast<System::Drawing::Icon^     >(resources-
>GetObject(L"$this.Icon")));
            this->KeyPreview = true;
            this->MainMenuStrip = this->menuStrip;
            this->MinimumSize = System::Drawing::Size(800, 600);
            this->Name = L"MainForm";
            this->Text = L"Forest";
            this->FormClosing                          +=                    gcnew
System::Windows::Forms::FormClosingEventHandler(this,
&MainForm::MainForm_FormClosing);
            this->FormClosed                           +=                    gcnew
System::Windows::Forms::FormClosedEventHandler(this, &MainForm::MainForm_FormClosed);
```

```
            this->Load          +=          gcnew          System::EventHandler(this,
&MainForm::MainForm_Load);
            this->Shown          +=          gcnew          System::EventHandler(this,
&MainForm::MainForm_Shown);
            this->Resize         +=          gcnew          System::EventHandler(this,
&MainForm::MainForm_Resize);
            this->menuStrip->ResumeLayout(false);
            this->menuStrip->PerformLayout();
            this->toolStrip->ResumeLayout(false);
            this->toolStrip->PerformLayout();
            this->statusStrip->ResumeLayout(false);
            this->statusStrip->PerformLayout();
            this->workspacePanel->ResumeLayout(false);
            (cli::safe_cast<System::ComponentModel::ISupportInitialize^      >(this-
>mainCanvas))->EndInit();
            this->ResumeLayout(false);
            this->PerformLayout();
      }
#pragma endregion
#pragma region Реализация интерфейса IMainView
public:
      /*вернуть интерфейс окна навигатора*/
      virtual property INavigatorView^ NavigatorView
      {
            INavigatorView^ get()
            {
                  return navigatorForm;
            }
      }
      /*вернуть интерфейс окна настроек*/
      virtual property ISettingsView^ SettingsView
      {
            ISettingsView^ get()
            {
                  return settingsForm;
            }
      }
      /*вернуть интерыейс окна отчета*/
      virtual property IReportView^ ReportView
      {
            IReportView^ get()
            {
                  return reportForm;
            }
      }
      /*вернуть интерфейс окна отчета*/
      virtual property IPlot3DView^ Plot3DView
      {
            IPlot3DView^ get()
            {
                  return plot3DForm;
            }
      }
      /*событие, пользовательская запрос*/
      virtual event UserRequestEventHandler^ UserRequest;
      /*установить презентера*/
      virtual property Object^ Presenter
      {
            void set(Object^ obj)
            {
                  presenter = dynamic_cast<MainPresenter^>(obj);
```

```
		}
	}
	/*выставить состояние формы "Занят выполнение асинхронной операции"
	asincOperationInfo - информация о выпоняемой операции*/
	virtual System::Void SetBusyStyle(String^ asincOperationInfo)
	{
		if(this->InvokeRequired == true)
		{
			this->Invoke(setBusyStyle,asincOperationInfo);
		}
		else
		{
			setBusyStyleMethod(asincOperationInfo);
		}
	}
	/*сбросить состояние формы "Занят выполнение асинхронной операции"*/
	virtual System::Void ResetBusyStyle()
	{
		if(this->InvokeRequired == true)
		{
			this->Invoke(resetBusyStyle);
		}
		else
		{
			resetBusyStyleMethod();
		}
	}
	/*выставляет графические данные*/
	virtual property IResultData^ GraphicsData
	{
		System::Void set(IResultData^ val)
		{
			this->graphicsData = val;
			if(this->graphicsData->SourceData != nullptr)
			{
				this->imgToolStripComboBox->Items->Clear();
				this->imgToolStripComboBox->Items->AddRange(graphicsData-
>SourceData->StrImages);
				this->imgToolStripComboBox->SelectedItem   =   graphicsData-
>SourceData->ActiveStrImage;
			}
			/*забираем себе изображение*/
			if(this->MainImage != this->graphicsData->SourceData->ActiveItem)
			{
				this->MainImage      =      this->graphicsData->SourceData-
>ActiveItem;
			}
			/*устанавливаем инструментам*/
			currentToolState->GraphicsData = this->graphicsData;
			GC::Collect();
		}
	}
#pragma endregion Реализация интерфейса IMainView
#pragma region Реализация интерфейса IMainViewTools
	/*возвращаем пикчербокс*/
public:	virtual property Control^ Canva
	{
		Control^ get()
		{
			return mainCanvas;
		}
```

```
	}
public:	virtual property ToolStripStatusLabel^ ResultStatusLabel
	{
		ToolStripStatusLabel^ get()
		{
			return resultToolStripStatusLabel;
		}
	}
public: virtual property System::Windows::Forms::ContextMenuStrip^ ContextMenuControl
	{
		System::Windows::Forms::ContextMenuStrip^ get()
		{
			return contextMenuStrip;
		}
	}
public: virtual property IResultData^ ResultData
	{
		IResultData^ get()
		{
			return this->graphicsData;
		}
	}
#pragma endregion Реализация интерфейса IMainViewTools
	/*свойство изображение*/
private: property Image^ MainImage
	{
		Image^ get()
		{
			return mainImage;
		}
		void set(Image^ img)
		{
			if(img != nullptr)
			{
				/*выставить автоскрол родительского контейнера canvas*/
				dynamic_cast<ScrollableControl^>(mainCanvas->Parent)-
>AutoScroll = true;
				mainImage = img;
				/*выставить предельные размеры*/
				minSizeImage = Prodact(img->Size,0.01f); // 1%
				maxSizeImage = Prodact(img->Size,5.0f); // 500%
				deltaZoom = 0.2f;
				/*на все окно*/
				fitCanvasSize(mainCanvas);
				/*отдать изображение навигатору*/
				navigatorForm->NavigatorImage = img;
			}
			else
			{
				mainImage = nullptr;
				mainCanvas->Location = Drawing::Point(0,0);
				mainCanvas->Size = Drawing::Size(2,2);
				navigatorForm->NavigatorImage = nullptr;
				this->Invalidate();
			}
			//GC::Collect();
		}
	}
		/*выставить по размеру контейнера*/
private: System::Void fitCanvasSize(System::Windows::Forms::Control^ canvasControl)
		{
```

```
            if(mainImage == nullptr)
                  return;
            Drawing::Size parentControlSize = canvasControl->Parent->Size;
            Single ch = Single(mainImage->Height)/(parentControlSize.Height);
            Single cw = Single(mainImage->Width)/(parentControlSize.Width);
            Drawing::Size newSize;
            if(ch>=cw) // приводим к высоте
            {
                  newSize.Height = parentControlSize.Height;
                  newSize.Width = Convert::ToInt32(mainImage->Width/ch);
            }
            else // приводим к ширине
            {
                  newSize.Width = parentControlSize.Width;
                  newSize.Height = Convert::ToInt32(mainImage->Height/cw);
            }
            zoomCanvasSize(canvasControl,newSize);
            }
            /*масштабировать +/- задается sing*/
private:          System::Void         zoomCanvasSize(System::Windows::Forms::Control^
canvasControl,Int32 sign)
      {
            if(mainImage == nullptr)
                  return;
            Int32      newWidth      =      canvasControl->Width+sign*canvasControl-
>Width*deltaZoom;
            Int32      newHeight       =       Convert::ToSingle(newWidth)/mainImage-
>Width*mainImage->Height;
            System::Drawing::Size                        newSize                        =
System::Drawing::Size(newWidth,newHeight);
            zoomCanvasSize(canvasControl,newSize);
      }
            /*масштабирование и выравние согласно старой позиции скрола*/
private:          System::Void         zoomCanvasSize(System::Windows::Forms::Control^
canvasControl,Drawing::Size newSize)
            {
                  /*проверка на возможность изменения размера у инструментов*/
            if(currentToolState->CanResizeControl == false)
            {
                  return;
            }
            /*проверка на допустимый размер*/
            if(newSize < minSizeImage)
            {
                  newSize = minSizeImage;
            }
            else if(newSize > maxSizeImage)
            {
                  newSize = maxSizeImage;
            }
            /*прокуручивающийся родитель*/
            ScrollableControl^                     parentControl                     =
dynamic_cast<ScrollableControl^>(canvasControl->Parent);
            /*старые размеры картинки*/
            Drawing::Size oldSize = canvasControl->Size;
            /*коэффициент масштабирования*/
            Single scaleFactor = Convert::ToSingle(newSize.Width)/oldSize.Width;
            /*старые скролы изображения*/
            Point oldScrollPosition = parentControl->AutoScrollPosition;
            /*центральная точка старого размера*/
            Single old_cX = oldScrollPosition.X - parentControl->Size.Width/2.0f;
```

```
            Single old_cY = oldScrollPosition.Y - parentControl->Size.Height/2.0f;
            /*позиционируем в центре родителя*/
            canvasControl->Visible = false;
            canvasControl->Size = newSize;
            centralPosition(canvasControl);
            canvasControl->Visible = true;
            /*центральная точка нового размера*/
            Single new_cX = -old_cX*scaleFactor;
            Single new_cY = -old_cY*scaleFactor;
            /* востановить положение прокрутки */
            parentControl->AutoScrollPosition =  Drawing::Point(Math::Round(new_cX -
parentControl->Size.Width/2.0f),
                  Math::Round(new_cY - parentControl->Size.Height/2.0f));
            /*обновить навигатор*/
            updateNavigatorRectangle();
            /*обновить статус масштаба*/
            updateZoomStatus();
            }
private: System::Void updateZoomStatus()
{
if(mainImage != nullptr)
{
      Single  zoomPercent  =  Convert::ToSingle(mainCanvas->Height)*100.0f/mainImage-
>Height;
      zoomToolStripStatusLabel->Text = zoomPercent.ToString("F1") + " %";
      zoomToolStripStatusLabel->ToolTipText       =       "Масштаб:       "       +
zoomToolStripStatusLabel->Text;
}
}
            /*привести к полному размеру*/
private: System::Void fullCanvasSize(System::Windows::Forms::Control^ canvasControl)
      {
            if(mainImage == nullptr)
                  return;
            Drawing::Size    newSize    =    Drawing::Size(mainImage->Width,mainImage-
>Height);
            zoomCanvasSize(canvasControl,newSize);
      }
            /*расположить по центру относительно родительского контейнера*/
private: System::Void centralPosition(System::Windows::Forms::Control^ canvasControl)
            {
            if(mainImage == nullptr)
                  return;
            ScrollableControl^                    parentControl                    =
dynamic_cast<ScrollableControl^>(canvasControl->Parent);
            if(canvasControl->Size < parentControl->ClientSize)
            {
                  Int32 X = (parentControl->ClientSize.Width - canvasControl->Width
+ parentControl->AutoScrollPosition.X)>>1;
                  Int32  Y  =  (parentControl->ClientSize.Height  -  canvasControl-
>Height + parentControl->AutoScrollPosition.Y)>>1;
                  canvasControl->Location = Point(X,Y);
            }
            else if (canvasControl->Size >= parentControl->ClientSize)
            {
                  Int32 X = parentControl->AutoScrollPosition.X;
                  Int32 Y = parentControl->AutoScrollPosition.Y;
                  canvasControl->Location = Point(X,Y);
            }
            else if (canvasControl->Width < parentControl->ClientSize.Width)
            {
```

```
                    Int32 X = (parentControl->ClientSize.Width - canvasControl->Width
+ parentControl->AutoScrollPosition.X)>>1;
                    Int32 Y = parentControl->AutoScrollPosition.Y;
                    canvasControl->Location = Point(X,Y);
             }
             else if (canvasControl->Height < parentControl->ClientSize.Height)
             {
                    Int32 X = parentControl->AutoScrollPosition.X;
                    Int32  Y  =  (parentControl->ClientSize.Height  -  canvasControl-
>Height + parentControl->AutoScrollPosition.Y)>>1;
                    canvasControl->Location = Point(X,Y);
             }
             }
             /*рисуем изображение*/
private: System::Void drawMainImage(Graphics^  graphics)
             {
             if(mainImage == nullptr)
                    return;
             //if(mainCanvas->Size != mainImage->Size)
             //{
                    /* если размер кавны больше размера изображения выставляем худшую
интерполяцию
                    чтобы видеть пикселы изображения*/
                    graphics->InterpolationMode                                    =
Drawing2D::InterpolationMode::NearestNeighbor;
             //}
             graphics->DrawImage(mainImage,mainCanvas->ClientRectangle);
             /*востановить интерполяцию*/
             graphics->InterpolationMode = Drawing2D::InterpolationMode::Default;
             }
private:   System::Void   navigatorToolStripMenuItem_Click(System::Object^     sender,
System::EventArgs^  e)
             {
             if(navigatorForm == nullptr)
                    return;
             navigatorForm->Visible = !navigatorForm->Visible;
             }
#pragma region обработчики событий кнопок zoom
private:    System::Void    zoom_inToolStripButton_Click(System::Object^       sender,
System::EventArgs^  e)
             {
                    zoomCanvasSize(mainCanvas,1);
             }
private:    System::Void    zoom_outToolStripButton_Click(System::Object^      sender,
System::EventArgs^  e)
             {
                    zoomCanvasSize(mainCanvas,-1);
             }
private:    System::Void    zoom_fitToolStripButton_Click(System::Object^      sender,
System::EventArgs^  e)
             {
                    fitCanvasSize(mainCanvas);
             }
private:    System::Void    zoom_100ToolStripButton_Click(System::Object^      sender,
System::EventArgs^  e)
             {
                    fullCanvasSize(mainCanvas);
             }
#pragma endregion обработчики событий кнопок zoom
private:           System::Void          navigatorForm_NavigatorRectangleMove(Object^
sender,Drawing::PointF delta)
```

```
            {
                  Single  sX  =  workspacePanel->ClientSize.Width  /  navigatorForm-
>NavigatorRectangle.Width;
                  Single  sY  =  workspacePanel->ClientSize.Height  /  navigatorForm-
>NavigatorRectangle.Height;
                  workspacePanel->AutoScrollPosition                              =
Drawing::Point(sX*delta.X,sY*delta.Y);
                  updateNavigatorRectangle();
            }
private:              System::Void             navigatorForm_NeedGraphicsData(Object^
sender,System::EventArgs^ e)
            {
                  navigatorForm->DrawingElement = currentToolState->DrawingElement;
            }
private:    System::Void    navigatorForm_VisibleChanged(System::Object^       sender,
System::EventArgs^  e)
            {
                  /*установить   видимость   Checked   свойство   кнопки,   согласно
видимости формы*/
                  navigatorToolStripMenuItem->Checked                             =
dynamic_cast<System::Windows::Forms::Form^>(sender)->Visible;
            }
private:    System::Void    settingsForm_VisibleChanged(System::Object^       sender,
System::EventArgs^  e)
            {
                  /*установить   видимость   Checked   свойство   кнопки,   согласно
видимости формы*/
                  settingsToolStripMenuItem->Checked                              =
dynamic_cast<System::Windows::Forms::Form^>(sender)->Visible;
            }
private:   System::Void   settingsToolStripMenuItem_Click(System::Object^      sender,
System::EventArgs^  e)
            {
            if(settingsForm == nullptr)
                  return;
            settingsForm->Visible = !settingsForm->Visible;

            }
private: System::Void updateNavigatorRectangle(void)
            {
                  if(mainImage == nullptr)
                        return;
                  ScrollableControl^                parentControl                 =
dynamic_cast<ScrollableControl^>(mainCanvas->Parent);
                  System::Drawing::Rectangle rect;
                  if(mainCanvas->ClientSize <= parentControl->ClientSize)
                  {
                        /*кавна меньше родителя*/
                        rect = mainCanvas->ClientRectangle;
                  }
                  else if(mainCanvas->ClientSize >= parentControl->ClientSize)
                  {
                        /*кавна больше родителя*/
                        rect      =      System::Drawing::Rectangle(-parentControl-
>AutoScrollPosition,parentControl->ClientSize);
                  }
                  else    if(mainCanvas->ClientSize.Height    >=    parentControl-
>ClientSize.Height)
                  {
                        /*кавна больше по высоте*/
                        rect = System::Drawing::Rectangle(0,
```

```
                                -parentControl->AutoScrollPosition.Y,
                                mainCanvas->ClientSize.Width,
                                parentControl->ClientSize.Height);
                        }
                        else    if(mainCanvas->ClientSize.Width    >=    parentControl-
>ClientSize.Width)
                        {
                                rect     =     System::Drawing::Rectangle(-parentControl-
>AutoScrollPosition.X,
                                0,
                                parentControl->ClientSize.Width,
                                mainCanvas->ClientSize.Height);
                        }
                        /*приводим к размеру навигатора*/
                        Drawing::PointF                  scalePointF                  =
Drawing::PointF(Convert::ToSingle(navigatorForm->NavigatorWidth)/mainCanvas->Width,
                                Convert::ToSingle(navigatorForm-
>NavigatorHeight)/mainCanvas->Height);
                        navigatorForm->NavigatorRectangle = rect*scalePointF;
                        //mainCanvas->Refresh();
                        mainCanvas->Invalidate();

                }
private:    System::Void    workspacePanel_Scroll(System::Object^          sender,
System::Windows::Forms::ScrollEventArgs^  e)
                {
                        updateNavigatorRectangle();
                }
private:   System::Void   reportToolStripMenuItem_Click(System::Object^      sender,
System::EventArgs^  e)
                {
                        reportForm->ShowDialog();
                        //GC::Collect();
                }
private:   System::Void   plot3DToolStripMenuItem_Click(System::Object^      sender,
System::EventArgs^  e)
                {
                        if(plot3DForm == nullptr)
                                return;
                        plot3DForm->Visible = !plot3DForm->Visible;
                }
private:    System::Void    plot3DForm_VisibleChanged(System::Object^        sender,
System::EventArgs^  e)
                {
                        /*установить   видимость   Checked   свойство   кнопки,   согласно
видимости формы*/
                        plot3DToolStripMenuItem->Checked                                =
dynamic_cast<System::Windows::Forms::Form^>(sender)->Visible;
                }
private:    System::Void    openToolStripMenuItem_Click(System::Object^       sender,
System::EventArgs^  e)
                {
                        //отправить сообщение слушающему презентеру
                        this->UserRequest(this,gcnew UrOpenImage());
                }

#pragma region обработчики событий pictureBox
private:      System::Void      mainCanvas_Paint(System::Object^            sender,
System::Windows::Forms::PaintEventArgs^  e)
                {
                        /*рисуем изображение*/
```

```
                drawMainImage(e->Graphics);
                /*рисуем данные в зависимости от состояния*/
                currentToolState->Canvas_Paint(sender,e);
                /*обновим навигатор*/
                navigatorForm->Refresh();
            }
private:     System::Void     mainCanvas_MouseClick(System::Object^          sender,
System::Windows::Forms::MouseEventArgs^  e)
            {
                currentToolState->Canvas_MouseClick(sender,e);
            }
private:     System::Void     mainCanvas_MouseDown(System::Object^          sender,
System::Windows::Forms::MouseEventArgs^  e)
            {
                currentToolState->Canvas_MouseDown(sender,e);
            }
private:      System::Void      mainCanvas_MouseUp(System::Object^           sender,
System::Windows::Forms::MouseEventArgs^  e)
            {
                currentToolState->Canvas_MouseUp(sender,e);
            }
private:     System::Void     mainCanvas_MouseMove(System::Object^          sender,
System::Windows::Forms::MouseEventArgs^  e)
            {
                currentToolState->Canvas_MouseMove(sender,e);
            }
#pragma endregion обработчики событий pictureBox
#pragma region выбор инструмента
            /*создать инструмент ручная калибровка*/
private:   System::Void   tCalibrateToolStripItem_Click(System::Object^      sender,
System::EventArgs^  e)
            {
                if(currentToolState != nullptr)
                {
                    if(currentToolState->GetType()-
>Equals(CalibrateToolState::typeid) == true )
                        return;
                    removeToolsEvents(currentToolState);
                    delete currentToolState;
                }
                currentToolState = gcnew CalibrateToolState(this);
                calibrateToolStripButton->Checked = true;
                editToolStripButton->Checked     =     !calibrateToolStripButton-
>Checked;
                addToolsEvents(currentToolState);
                updateToolsStatus(currentToolState,nullptr);
                if(this->graphicsData != nullptr)
                {
                    currentToolState->GraphicsData = this->graphicsData;
                }
            }
            /*создать инструмент редактор*/
private:    System::Void    tEditToolStripItem_Click(System::Object^         sender,
System::EventArgs^  e)
            {
                if(currentToolState != nullptr)
                {
                    if(currentToolState->GetType()-
>Equals(EditToolState::typeid) == true )
                        return;
                    removeToolsEvents(currentToolState);
```

```
                        delete currentToolState;
                }
                currentToolState = gcnew EditToolState(this);
                editToolStripButton->Checked = true;
                calibrateToolStripButton->Checked     =     !editToolStripButton-
>Checked;
                addToolsEvents(currentToolState);
                updateToolsStatus(currentToolState,nullptr);
                if(this->graphicsData != nullptr)
                {
                        currentToolState->GraphicsData = this->graphicsData;
                }
        }
private: System::Void updateToolsStatus(Object^ sender,System::EventArgs^ e)
      {
            toolsToolStripStatusLabel->Text  =  dynamic_cast<BaseToolState^>(sender)-
>Name;
            /*toolsToolStripStatusLabel->Image                                    =
dynamic_cast<BaseToolState^>(sender)->ToolImage;*/
            toolsToolStripStatusLabel->Image = (editToolStripButton->Checked == true
? editToolStripButton->Image : calibrateToolStripButton->Image );
            toolsToolStripStatusLabel->ToolTipText   =   "Инструмент   "   +   "\""   +
toolsToolStripStatusLabel->Text + "\"";
      }
            /*подписываемся на события BaseToolState*/
private: System::Void addToolsEvents(BaseToolState^ cts)
            {
                  cts->UserRequest                    +=                    gcnew
UserRequestEventHandler(this,&MainForm::currentToolState_UserRequest);
            }
            /*удаляем подписку на события BaseToolState*/
private: System::Void removeToolsEvents(BaseToolState^ cts)
            {
                  cts->UserRequest                    -=                    gcnew
UserRequestEventHandler(this,&MainForm::currentToolState_UserRequest);
            }
#pragma endregion выбор инструмента
#pragma region обработчики событий инструментов главной формы
            /*обработчик события отправки пользовательской запрос*/
private:   System::Void   currentToolState_UserRequest(Object^   sender,IUserRequest^
userRequest)
      {
            /*перенаправляем запрос презентеру*/
            this->UserRequest(this,userRequest);
      }
#pragma endregion обработчики событий инструментов главной формы
#pragma region обработчики событий таймера подсказок
            /*таймер отображения подсказок*/
private:     System::Void     timerForToolTip_Tick(System::Object^          sender,
System::EventArgs^  e)
            {
                  /*запрашиваем подсказку у инструмента*/
                  if(currentToolState != nullptr)
                  {
                        String^ toolTipText = currentToolState->ToolTipText;
                        if(toolTipText != nullptr)
                        {
                        toolTip->SetToolTip(mainCanvas,toolTipText);
                        return;
                        }
                  }
```

```
                toolTip->RemoveAll();
            }
#pragma endregion таймер подсказок
#pragma region обработчики событий главной формы
private:      System::Void      MainForm_FormClosed(System::Object^           sender,
System::Windows::Forms::FormClosedEventArgs^  e) {
            try
            {
                helpProcess->Kill();
            }
            catch(Exception^)
            {
            }
      }
            /*загрузка главной формы*/
private: System::Void MainForm_Load(System::Object^  sender, System::EventArgs^  e)
            {
                timerForToolTip->Enabled = true;
                /*применяем настройки*/
                switch (mainFormSettings->MainFormWindowState)
                {
                case FormWindowState::Normal:
                    this->WindowState = FormWindowState::Normal;
                    this->Location = mainFormSettings->MainFormLocation;
                    this->Size = mainFormSettings->MainFormSize;
                    break;
                case FormWindowState::Minimized:
                    this->WindowState = FormWindowState::Maximized;
                    break;
                case FormWindowState::Maximized:
                    this->WindowState = FormWindowState::Maximized;
                    break;
                }
                /*просим данные у модели*/
                this->UserRequest(this,gcnew UrUpdateResultData);
            }
            /*отбражение фомы первый раз*/
private: System::Void MainForm_Shown(System::Object^  sender, System::EventArgs^  e)
            {
                /*показываем  все  окна  (формы  покажутся  в  зависимости  от
настроек)*/
                navigatorForm->Visible = mainFormSettings->NavigatorFormVisible;
                plot3DForm->Visible = mainFormSettings->Plot3DFormVisible;
                settingsForm->Visible = mainFormSettings->SettingsFormVisible;
            }
            /*закрытие формы*/
private:      System::Void      MainForm_FormClosing(System::Object^           sender,
System::Windows::Forms::FormClosingEventArgs^  e)
            {
            /*сохраняем настройки*/
        mainFormSettings->MainFormWindowState = this->WindowState;
            mainFormSettings->MainFormLocation = this->Location;
            mainFormSettings->MainFormSize = this->Size;
            mainFormSettings->NavigatorFormVisible = navigatorForm->Visible;
            mainFormSettings->Plot3DFormVisible = plot3DForm->Visible;
            mainFormSettings->SettingsFormVisible = settingsForm->Visible;
            mainFormSettings->Save();
            /*принудитеотное закрытие формы отчета*/
            if(reportForm != nullptr)
            {
                reportForm->Close();
```

```
            }
            }
      /*прокрутка колеса главной фомы*/
private:          System::Void          MainForm_MouseWheel(Object^          sender,
System::Windows::Forms::MouseEventArgs^ e)
            {
                  if(e->Delta != 0)
                  {
                        Int32 sign = e->Delta/Math::Abs(e->Delta);
                        if(sign > 0)
                        {
                              zoom_inToolStripButton_Click(this,nullptr);
                        }
                        else
                        {
                              zoom_outToolStripButton_Click(this,nullptr);
                        }
                  }
            }
      /*изменения размеров главной фомы*/
private: System::Void MainForm_Resize(System::Object^  sender, System::EventArgs^  e)
            {
                  centralPosition(mainCanvas);
                  updateNavigatorRectangle();
            }
#pragma endregion обработчики событий главной формы
private:    System::Void    runToolStripButton_Click(System::Object^         sender,
System::EventArgs^  e)
            {
                  /*при запуске переключаемся на редактор*/
                  this->tEditToolStripItem_Click(nullptr,nullptr);
                  this->UserRequest(this,gcnew UrRunAlgoritm());
            }
#pragma region ф. изменения стиля главной формы
private: System::Void setBusyStyleMethod(String^ info)
      {
            toolStripProgressBar->MarqueeAnimationSpeed = 5;
            toolStripProgressBar->Style = ProgressBarStyle::Marquee;
            infoToolStripStatusLabel->Text = info;
            this->UseWaitCursor = true;
            this->Refresh();
      }
private: System::Void resetBusyStyleMethod()
            {
                  toolStripProgressBar->Style = ProgressBarStyle::Blocks;
                  infoToolStripStatusLabel->Text = "";
                  this->UseWaitCursor = false;
                  this->Refresh();
            }
#pragma endregion ф. изменения стиля главной формы
//    private:    System::Void    toolStripButton1_Click(System::Object^       sender,
System::EventArgs^  e) {
//                   this->UserRequest(this,gcnew UrRecalculate());
//             }
#pragma region обработчики событий combobox
private:    System::Void    imgToolStripComboBox_SelectedIndexChanged(System::Object^
sender, System::EventArgs^  e)
            {
                  /*если имена не совпадают отправляем запрос на сену картинки*/
                  if(imgToolStripComboBox->SelectedItem      !=      graphicsData-
>SourceData->ActiveStrImage)
```

```
                  {
                        this->UserRequest(this,gcnew
UrChangeImage(imgToolStripComboBox->SelectedItem->ToString()));
                  }
                  imgToolStripComboBox->SelectedItem   =   graphicsData->SourceData-
>ActiveStrImage;
                  /*сбровим активный контрол формы, что бы comboBox терял фокус*/
                  this->ActiveControl = nullptr;
            }
            /*обработчик необходимости рисования чего нить в комбобоксе*/
private:             System::Void             imgToolStripComboBox_MeasureItem(Object^
sender,System::Windows::Forms::MeasureItemEventArgs^ e)
            {
            e->ItemHeight = cbItemHeight;
            }
#pragma endregion обработчики событий combobox
            /*рисование изображений в combobox*/
private:       System::Void       imgToolStripComboBox_DrawItem(Object^       sender,
System::Windows::Forms::DrawItemEventArgs^ e)
            {
                  Image^ img;
                  if((img       =       graphicsData->SourceData->Item[graphicsData-
>SourceData->StrImages[e->Index]]) != nullptr )
                  {
                  e->Graphics->DrawImage(img,e->Bounds);
                  }
                  else
                  {
                        e->Graphics->FillRectangle(gcnew
SolidBrush(System::Drawing::SystemColors::AppWorkspace),e->Bounds);
                  }
                  if(e->State == DrawItemState::Selected)
                  {
                        Int32 w_ = 4;
                        Drawing::Rectangle rect_ = e->Bounds;
                        rect_.Inflate(-w_>>1,-w_>>1);
                        e->Graphics->DrawRectangle(gcnew
Pen(Color::FromArgb(150,Color::Red),w_),rect_);
                  }
            }
private:    System::Void    helpToolStripButton_Click(System::Object^        sender,
System::EventArgs^  e)
            {
                  try
                  {
                        if(helpProcess->HasExited   ==   true)   //если   завершен,
запускаем заново
                        {
                              helpProcess->Start();
                        }
                  }
                  catch (Exception^ )
                  {
                        try
                        {
                              helpProcess->Start();
                        }
                        catch(Exception^)
                        {
                        }
                  }
```

```
		}
private:	System::Void	aboutToolStripMenuItem_Click(System::Object^		sender,
System::EventArgs^	e)
		{
			AboutForm^ aboutForm = gcnew AboutForm();
			aboutForm->ShowDialog(this);
		}
};
}
```

Model.h

```
#pragma once
#include "IModelImageStrategy.h"
#include "OneImageModel.h"
#include "TwoImageModel.h"
#include "IModel.h"
#include "ModelSettings.h"
#include "AsyncOperation.h"
namespace Forest {
	using namespace System;
	using namespace System::ComponentModel;
	using namespace System::Collections;
	using namespace System::Windows::Forms;
	using namespace System::Data;
	using namespace System::Drawing;
	using namespace System::Drawing::Drawing2D;
	using namespace System::Reflection;
	using namespace Forest::SavedSettings;
/*Класс модели, должен реализовать интерфейс (IModel) для презенторов*/
ref class Model: public IModel
{
public:
	Model(void)
	{
		/*настройки*/
		this->modelSettings = gcnew SavedSettings::ModelSettings();
		/*загрузить модель*/
		this->ImageStrategies = modelSettings->ImageStrategies;
		/*создать себе экземпляр AsincOperation и подписаться на его события*/
		asyncOperation = AsyncOperation::InstanceOperation;
		asyncOperation->StartAsinc		+=		gcnew
AsincNotificationEventHandler(this,&Model::asyncOperation_StartAsinc);
		asyncOperation->StopAsinc		+=		gcnew
AsincNotificationEventHandler(this,&Model::asyncOperation_StopAsinc);
	}
protected:
	~Model()
	{
		/*сохранить состояние из текущего*/
		modelSettings->ImageStrategies = ImageStrategies;
		modelSettings->Save();
		/*обнулить текущее*/
		modelImageStrategy = nullptr;
		/*обнулить AsincOperation*/
		asyncOperation->StartAsinc		-=		gcnew
AsincNotificationEventHandler(this,&Model::asyncOperation_StartAsinc);
		asyncOperation->StopAsinc		-=		gcnew
AsincNotificationEventHandler(this,&Model::asyncOperation_StopAsinc);
		asyncOperation = nullptr;
```

```
	}
private:
	/*класс реализует логику работы программы
	  сделан через интерфейс для динамической подмены
	  поведения в зависимости от выбранного алгоритма
	  одна или две картинки*/
	IModelImagesStrategy^ modelImageStrategy;
	/*класс настройки*/
	ModelSettings^ modelSettings;
	/*асинхронная операция*/
	AsyncOperation^ asyncOperation;
	/*подписываемся на события IModelImagesStrategy*/
	System::Void addImageStrategyEvents(IModelImagesStrategy^ mis)
	{
		modelImageStrategy->PropertyChanged		+=		gcnew
System::EventHandler(this,&Model::modelImageStrategy_PropertyChanged);
		modelImageStrategy->ImageStrategyChanged		+=		gcnew
ImageStrategyEventHandler(this,&Model::ImageStrategies::set);
		modelImageStrategy->RequestExecuted		+=		gcnew
System::EventHandler(this,&Model::modelImageStrategy_RequestExecuted);
	}
	/*удаляем подписку на события IModelImagesStrategy*/
	System::Void removeImageStrategyEvents(IModelImagesStrategy^ mis)
	{
		/*программно изменилиси свойства модели*/
		modelImageStrategy->PropertyChanged		-=		gcnew
System::EventHandler(this,&Model::modelImageStrategy_PropertyChanged);
		/*изменилась стратегия модели (изображения)*/
		modelImageStrategy->ImageStrategyChanged		-=		gcnew
ImageStrategyEventHandler(this,&Model::ImageStrategies::set);
		/*уведомление о выполненном запросе*/
		modelImageStrategy->RequestExecuted		-=		gcnew
System::EventHandler(this,&Model::modelImageStrategy_RequestExecuted);
	}
#pragma region обработчики событий asyncOperation
	System::Void asyncOperation_StartAsinc(Object^ sender,Object^ args)
	{
		AsyncOperation^ ao = dynamic_cast<AsyncOperation^>(sender);
		/*отправить слушающему презентеру*/
		this->StartAsinc(this,ao->OperationInfo);

	}
	System::Void asyncOperation_StopAsinc(Object^ sender,Object^ args)
	{
		AsyncOperation^ ao = dynamic_cast<AsyncOperation^>(sender);
		/*отправить слушающему презентеру*/
		this->StopAsinc(this,nullptr);
	}
#pragma endregion обработчики событий asyncOperation
	/*свойство MODEL_IMAGE_STRATEGIES*/
	[BrowsableAttribute(false)]
	property MODEL_IMAGE_STRATEGIES ImageStrategies
	{
		MODEL_IMAGE_STRATEGIES get()
		{
			return modelImageStrategy->ImageStrategies;
		}
		System::Void set(MODEL_IMAGE_STRATEGIES val)
		{
			if(modelImageStrategy != nullptr)
			{
```

```
                        /*отпишемся от событий modelImageStrategy*/
                        removeImageStrategyEvents(modelImageStrategy);
                    }
                    /*сохранить предыдущую стратегию*/
                    IModelImagesStrategy^ temp_mis = modelImageStrategy;
                    /*загружаем нужную модель*/
                    switch (val)
                    {
                    case MODEL_IMAGE_STRATEGIES::OneImg:
                        modelImageStrategy = gcnew OneImageModel();
                        break;
                    case MODEL_IMAGE_STRATEGIES::TwoImg:
                        modelImageStrategy = gcnew TwoImageModel();
                        break;
                    }
                    /*подписываемся на события modelImageStrategy*/
                    addImageStrategyEvents(modelImageStrategy);
                    /*событируем о подмене модели в настройках*/
                    this->ReplaceProperty(temp_mis,this->modelImageStrategy);
                    if(temp_mis != nullptr)
                    {
                        delete temp_mis;
                        temp_mis = nullptr;
                    }
                    /*просим модель обновить данные (обновит и отправит данные на
форму)*/
                    modelImageStrategy->RequestToExecute(gcnew MrUpdateResultData);
                    //GC::Collect();
                }
            }
#pragma region реализация интерфейса IModel
public:
/*событие подмены объекта настроек*/
virtual event ReplacePropertyHandler^ ReplaceProperty;
/*событие изменились данные настроек*/
virtual event System::EventHandler^ PropertyChanged;
/*получить список объектов настроек*/
virtual List<Object^>^ GetSettingObjects()
{
    return modelImageStrategy->GetSettingObjects();
}
/*запрос на выполнение*/
virtual System::Void RequestToExecute(IModelRequest^ modelRequest)
{
    //уведометь выбранную стратегию
    modelImageStrategy->RequestToExecute(modelRequest);
}
virtual event System::EventHandler^ RequestExecuted;
/*события об асинхронных операциях*/
virtual event AsincNotificationEventHandler^ StartAsinc;
virtual event AsincNotificationEventHandler^ StopAsinc;
#pragma endregion реализация интерфейса IModel
#pragma region обработчики событий стратегии модели
private:        System::Void        modelImageStrategy_PropertyChanged(Object^
sender,System::EventArgs^ e)
            {
                this->PropertyChanged(this,nullptr);
            }
private:        System::Void        modelImageStrategy_RequestExecuted(Object^
sender,System::EventArgs^ e)
            {
```

```
                this->RequestExecuted(sender,nullptr);
            }
#pragma endregion обработчики событий стратегии модели
};
}
```

Methods.cpp

```
#include "stdafx.h"
#include "Methods.h"
namespace CubaturnicLib {
	/*Methods*/
	Single Methods::FrustumMethod(Single d,Single D,Single L)
	{
		if(d <= 0.0f)
		{
			throw gcnew Exception("Недопустимое значение верхнего диаметра d:
" + d.ToString());
		}
		if(D <= 0.0f)
		{
			throw gcnew Exception("Недопустимое значение нижнего диаметра D:
" + D.ToString());
		}
		if(L <= 0.0f)
		{
			throw gcnew Exception("Недопустимое значение длины L: " +
L.ToString());
		}
		Single d_sq = d*d;
		Single D_sq = D*D;
		Single d_D = d*D;
		return (float)((L*(d_sq+D_sq+d_D))*fm_const);
	}
	Single Methods::TopDiameterMeanRiseMethod(Single d,Single s,Single L)
	{
		if(d <= 0.0f)
		{
			throw gcnew Exception("Недопустимое значение верхнего диаметра d:
" + d.ToString());
		}
		if(s < 0.0f)
		{
			throw gcnew Exception("Недопустимое значение сбега бревна s: " +
s.ToString());
		}
		if(L <= 0.0f)
		{
			throw gcnew Exception("Недопустимое значение длины L: " +
L.ToString());
		}
		Single dc = Converter::TopSectionToMidSection(d,L,s);
		Single dc_sq = dc*dc;
		return (float)((L*dc_sq)*tdmrm_const);
	}
	Single Methods::MidSectionMethod(Single dc,Single L)
	{
		if(dc <= 0.0f)
		{
```

```
                throw gcnew Exception("Недопустимое значение срединного диаметра
dc: " + dc.ToString());
            }
            if(L <= 0.0f)
            {
                throw gcnew Exception("Недопустимое значение длины L: " +
L.ToString());
            }
            Single dc_sq = dc*dc;
            return (float)((L*dc_sq)*msm_const);
      }
      Single Methods::EndSectionsMethod(Single d,Single D,Single L)
      {
            if(d <= 0.0f)
            {
                throw gcnew Exception("Недопустимое значение верхнего диаметра d:
" + d.ToString());
            }
            if(D <= 0.0f)
            {
                throw gcnew Exception("Недопустимое значение нижнего диаметра D:
" + D.ToString());
            }
            if(L <= 0.0f)
            {
                throw gcnew Exception("Недопустимое значение длины L: " +
L.ToString());
            }
            Single d_sq = d*d;
            Single D_sq = D*D;
            return (float)((L*(d_sq+D_sq))*esm_const);
      }
      Single Methods::TableGOST2708Method(String^ key_d,String^ key_L,Int32 num)
      {
            return GOST_2708::InstanceGOST->Table[num]->Value[key_L,key_d];
      }
      Single Methods::GetRise(Single d,Single D,Single L)
      {
            if(d <= 0.0f)
            {
                throw gcnew Exception("Недопустимое значение верхнего диаметра d:
" + d.ToString());
            }
            if(D <= 0.0f)
            {
                throw gcnew Exception("Недопустимое значение нижнего диаметра D:
" + D.ToString());
            }
            if(d > D)
            {
                throw gcnew Exception("Верхний диаметр не может быть больше
нижнего: " + D.ToString());
            }
            if(L <= 0.0f)
            {
                throw gcnew Exception("Недопустимое значение длины L: " +
L.ToString());
            }
            return (D-d)/L;
      }
}
```

NavigatorForm.h

```
#pragma once
#include "INavigatorView.h"
#include "Global.h"
#include "NavigatorPresenter.h"
#include "NavigatorFormSettings.h"
#include "Ellipse2D.h"
namespace Forest {
	using namespace System;
	using namespace System::Reflection;
	using namespace System::ComponentModel;
	using namespace System::Collections;
	using namespace System::Windows::Forms;
	using namespace System::Data;
	using namespace System::Drawing;
	using namespace Global;
	using namespace Forest::SavedSettings;
	using namespace Graphics2D;
	/// <summary>
	/// Сводка для NavigatorForm
	/// </summary>
	ref class NavigatorForm : public System::Windows::Forms::Form ,public
INavigatorView
	{
	public:
		/*control - копия которого будет отображаться в навигаторе*/
		NavigatorForm()
		{
			InitializeComponent();
			//
			//TODO: добавьте код конструктора
			//
			/*создание настроек*/
			navigatorFormSettings = gcnew NavigatorFormSettings();
			/*установка минимального размера канвы навигатора*/
			minSizeNavigatorCanvas = Drawing::Size(240,180);
			/*загрузка курсоров*/
			try
			{
				handCursor = gcnew
System::Windows::Forms::Cursor(Assembly::GetExecutingAssembly()-
>GetManifestResourceStream("hand.cur"));
				handCaptureCursor = gcnew
System::Windows::Forms::Cursor(Assembly::GetExecutingAssembly()-
>GetManifestResourceStream("hand_capture.cur"));
			}
			catch (Exception^ )
			{
				handCursor = nullptr;
				handCaptureCursor = nullptr;
			}
			/*устанвка курсороа "рука"*/
			setHandCursor();
		}
	protected:
		/// <summary>
		/// Освободить все используемые ресурсы.
		/// </summary>
		~NavigatorForm()
```

```
		{
			drawingElement = nullptr;
			if (components)
			{
				delete components;
			}
		}
#pragma region приватные поля
private:
	NavigatorPresenter^ presenter;
	/*изображение навигатора*/
	Image^ navigatorImage;
	/*прямоугольник навигатора*/
	Drawing::RectangleF navigatorRectangle;
	/*минимальные размеры окна наигатора*/
	Drawing::Size minSizeNavigatorCanvas;
	/*кавна PictureBox*/
	System::Windows::Forms::PictureBox^  navigatorCanvas;
	/*флаг - захват навигатора*/
	bool isNavigatorRectactangleCaptured;
	/*точка захвата прямоугольника*/
	Drawing::PointF pointOfCapture;
	/*курсоры*/
	System::Windows::Forms::Cursor^ handCursor;
	System::Windows::Forms::Cursor^ handCaptureCursor;
	/*настройки*/
	NavigatorFormSettings^ navigatorFormSettings;
	/*какието графические данные*/
	IGraphicElement^ drawingElement;
#pragma endregion приватные поля
	private:
		/// <summary>
		/// Требуется переменная конструктора.
		/// </summary>
		System::ComponentModel::Container ^components;
#pragma region Windows Form Designer generated code
		/// <summary>
		/// Обязательный метод для поддержки конструктора - не изменяйте
		/// содержимое данного метода при помощи редактора кода.
		/// </summary>
		void InitializeComponent(void)
		{
			System::ComponentModel::ComponentResourceManager^    resources   =
(gcnew System::ComponentModel::ComponentResourceManager(NavigatorForm::typeid));
			this->navigatorCanvas                    =                    (gcnew
System::Windows::Forms::PictureBox());
			(cli::safe_cast<System::ComponentModel::ISupportInitialize^
>(this->navigatorCanvas))->BeginInit();
			this->SuspendLayout();
			//
			// navigatorCanvas
			//
			this->navigatorCanvas->Location = System::Drawing::Point(0, 0);
			this->navigatorCanvas->Name = L"navigatorCanvas";
			this->navigatorCanvas->Size = System::Drawing::Size(50, 50);
			this->navigatorCanvas->TabIndex = 0;
			this->navigatorCanvas->TabStop = false;
			this->navigatorCanvas->Paint                +=                gcnew
System::Windows::Forms::PaintEventHandler(this,
&NavigatorForm::canvasNavigator_Paint);
```

```
                this->navigatorCanvas->MouseDown += gcnew
System::Windows::Forms::MouseEventHandler(this,
&NavigatorForm::navigatorCanvas_MouseDown);
                this->navigatorCanvas->MouseMove += gcnew
System::Windows::Forms::MouseEventHandler(this,
&NavigatorForm::navigatorCanvas_MouseMove);
                this->navigatorCanvas->MouseUp += gcnew
System::Windows::Forms::MouseEventHandler(this,
&NavigatorForm::navigatorCanvas_MouseUp);
                //
                // NavigatorForm
                //
                this->BackColor = System::Drawing::SystemColors::AppWorkspace;
                this->ClientSize = System::Drawing::Size(274, 210);
                this->Controls->Add(this->navigatorCanvas);
                this->DoubleBuffered = true;
                this->FormBorderStyle =
System::Windows::Forms::FormBorderStyle::FixedSingle;
                this->Icon = (cli::safe_cast<System::Drawing::Icon^  >(resources-
>GetObject(L"$this.Icon")));
                this->MaximizeBox = false;
                this->MinimizeBox = false;
                this->Name = L"NavigatorForm";
                this->ShowInTaskbar = false;
                this->Text = L"Навигатор";
                this->FormClosing += gcnew
System::Windows::Forms::FormClosingEventHandler(this,
&NavigatorForm::NavigatorForm_FormClosing);
                this->Load += gcnew System::EventHandler(this,
&NavigatorForm::NavigatorForm_Load);
                (cli::safe_cast<System::ComponentModel::ISupportInitialize^
>(this->navigatorCanvas))->EndInit();
                this->ResumeLayout(false);
        }
#pragma endregion
#pragma region реализация интерфейса INavigatorView
public:
    /*движение навигатора*/
    virtual event NavigatorRectangleEventHandler^ NavigatorRectangleMove;
    /*Установить изображение навигатора*/
    virtual property Image^ NavigatorImage
    {
        void set(Image^ img)
        {
            if(img != nullptr)
            {
                navigatorCanvas->Location = Drawing::Point(0,0);
                navigatorCanvas->Size = Drawing::Size(0,0);
                if( img->Size <= minSizeNavigatorCanvas )
                {
                    /*если картинка меньше минимального допустимого
размера навигатора
                     размер окна выставляем в minSizeNavigatorCanvas
                     картинку не масштабируем
                     позиционируем картинку по центру формы*/
                    this->ClientSize = minSizeNavigatorCanvas;
                    navigatorCanvas->Size = img->Size;
                    navigatorImage = gcnew Bitmap(img);
                    centralPosition(navigatorCanvas);
                }
                else
```

```
                        {
                              /*если картинка больше, масштабируем
                               клиентскую область окна подгоняем под картинку*/
                              Single          ch          =          Single(img-
>Height)/minSizeNavigatorCanvas.Height;
                              Single          cw          =          Single(img-
>Width)/minSizeNavigatorCanvas.Width;
                              if(ch>=cw) // приводим к высоте
                              {
                                    this->ClientSize                              =
Drawing::Size(Convert::ToInt32(img->Width/ch),minSizeNavigatorCanvas.Height);
                              }
                              else // приводим к ширине
                              {
                                    this->ClientSize                              =
Drawing::Size(minSizeNavigatorCanvas.Width,Convert::ToInt32(img->Height/cw));
                              }
                              navigatorCanvas->Size = this->ClientSize;
                              navigatorImage  =  gcnew  Bitmap(img,navigatorCanvas-
>Size);
                        }
                        /*инициализация прямоугольника */
                        navigatorRectangle                                        =
Drawing::RectangleF(Drawing::PointF(0.0f,0.0f),navigatorCanvas->Size);
                  }
                  else
                  {
                        navigatorImage = nullptr;
                        navigatorCanvas->Location = Drawing::Point(0,0);
                        navigatorCanvas->Size = Drawing::Size(0,0);
                        this->Size = minSizeNavigatorCanvas;
                  }
                  this->Refresh();

            }
      }
      virtual property Int32 NavigatorHeight
      {
            Int32 get(void)
            {
                  return navigatorCanvas->Height;
            }
      }
      virtual property Int32 NavigatorWidth
      {
            Int32 get(void)
            {
                  return navigatorCanvas->Width;
            }
      }
      virtual property Drawing::Size NavigatorSize
      {
            Drawing::Size get(void)
            {
                  return navigatorCanvas->Size;
            }
      }
      virtual property Drawing::RectangleF NavigatorRectangle
      {
            void set(Drawing::RectangleF rect)
            {
```

```
                    if(navigatorImage != nullptr)
                    {
                          navigatorRectangle = rect;
                          navigatorCanvas->Refresh();
                    }
             }
             Drawing::RectangleF get(void)
             {
                    return navigatorRectangle;
             }
      }
      virtual property Object^ Presenter
      {
             void set(Object^ obj)
             {
                    presenter = dynamic_cast<NavigatorPresenter^>(obj);
             }
      }
#pragma endregion реализация интерфейса INavigatorView
#pragma region публичные свойства
public:
      /*нужны графические данные*/
      event EventHandler^ NeedGraphicsData;
      /*данные, которые рисует*/
      property IGraphicElement^ DrawingElement
      {
             void set(IGraphicElement^ val)
             {
                    drawingElement = val;
             }
      }
#pragma endregion публичные свойства
private: System::Void centralPosition(System::Windows::Forms::Control^ canvasControl)
      {
             if(navigatorImage != nullptr)
             {
                    ScrollableControl^              scrollableControl              =
dynamic_cast<ScrollableControl^>(canvasControl->Parent);
                    if(canvasControl->Size < scrollableControl->ClientSize)
                    {
                          Int32    X    =    (scrollableControl->ClientSize.Width    -
canvasControl->Width + scrollableControl->AutoScrollPosition.X)/2;
                          Int32    Y    =    (scrollableControl->ClientSize.Height    -
canvasControl->Height + scrollableControl->AutoScrollPosition.Y)/2;
                          canvasControl->Location = Point(X,Y);
                    }
                    else if (canvasControl->Size >= scrollableControl->ClientSize)
                    {
                          Int32 X = scrollableControl->AutoScrollPosition.X;
                          Int32 Y = scrollableControl->AutoScrollPosition.Y;
                          canvasControl->Location = Point(X,Y);
                    }
                    else     if     (canvasControl->Width     <     scrollableControl-
>ClientSize.Width)
                    {
                          Int32    X    =    (scrollableControl->ClientSize.Width    -
canvasControl->Width + scrollableControl->AutoScrollPosition.X)/2;
                          Int32 Y = scrollableControl->AutoScrollPosition.Y;
                          canvasControl->Location = Point(X,Y);
                    }
```

```
                  else     if     (canvasControl->Height     <     scrollableControl-
>ClientSize.Height)
                  {
                        Int32 X = scrollableControl->AutoScrollPosition.X;
                        Int32    Y    =    (scrollableControl->ClientSize.Height    -
canvasControl->Height + scrollableControl->AutoScrollPosition.Y)/2;
                        canvasControl->Location = Point(X,Y);
                  }
            }
      }
private:     System::Void     canvasNavigator_Paint(System::Object^          sender,
System::Windows::Forms::PaintEventArgs^  e)
            {
                  drawNavigatorImage(e->Graphics);
                  drawDrawingElement(e->Graphics);
                  drawNavigatorRectangle(e->Graphics);
            }
            /*риссуем изображение*/
private: System::Void drawNavigatorImage(Graphics^  graphics)
            {
                  if(navigatorImage != nullptr)
                  {
                        graphics->DrawImage(navigatorImage,navigatorCanvas-
>ClientRectangle);
                  }
            }
private: System::Void drawNavigatorRectangle(Graphics^  graphics)
            {
                  if(navigatorImage != nullptr)
                  {
                        graphics->DrawRectangle(gcnew
Pen(Color::FromArgb(150,Color::Red),6.0f),

      navigatorRectangle.X,

      navigatorRectangle.Y,

      navigatorRectangle.Width,

      navigatorRectangle.Height);
                  }
            }
private: System::Void drawDrawingElement(Graphics^  graphics)
            {
                  if(navigatorImage != nullptr)
                  {
                        /*попроси данные*/
                        this->NeedGraphicsData(this,nullptr);
                        if(drawingElement != nullptr)
                        {
                              drawingElement->Draw(graphics,navigatorCanvas-
>ClientRectangle.Size);
                              graphics->ResetTransform();
                              drawingElement = nullptr;
                        }
                  }
            }
private:    System::Void    navigatorCanvas_MouseDown(System::Object^        sender,
System::Windows::Forms::MouseEventArgs^  e)
            {
                  if(navigatorImage != nullptr)
```

```
                {
                        if(e->Button                                                    ==
System::Windows::Forms::MouseButtons::Left)
                        {
                                if(navigatorRectangle.Contains(e->Location))
                                {
                                        isNavigatorRectactangleCaptured = true;
                                        setHandCaptureCursor();
                                        /*запомнить точку нажатия */
                                        pointOfCapture     =     PointF(e->Location.X-
navigatorRectangle.X,e->Location.Y-navigatorRectangle.Y);
                                        return;
                                }
                        }
                }
                isNavigatorRectactangleCaptured = false;
                setHandCursor();
        }
private:    System::Void    navigatorCanvas_MouseMove(System::Object^        sender,
System::Windows::Forms::MouseEventArgs^  e)
        {
                if(isNavigatorRectactangleCaptured == true)
                {
                        /*отправляем сообщение*/
                        NavigatorRectangleMove(this,Drawing::PointF(e->X         -
pointOfCapture.X , e->Y - pointOfCapture.Y));
                }
        }
private:    System::Void    navigatorCanvas_MouseUp(System::Object^         sender,
System::Windows::Forms::MouseEventArgs^  e)
        {
                isNavigatorRectactangleCaptured = false;
                setHandCursor();
        }
private: System::Void setHandCaptureCursor()
        {
                if(handCaptureCursor != nullptr)
                     navigatorCanvas->Cursor = handCaptureCursor;
        }
private: System::Void setHandCursor()
        {
                if(handCursor != nullptr)
                     navigatorCanvas->Cursor = handCursor;
        }
private:    System::Void    NavigatorForm_FormClosing(System::Object^        sender,
System::Windows::Forms::FormClosingEventArgs^  e)
        {
                if(e->CloseReason == CloseReason::UserClosing)
                {
                        e->Cancel = true;
                        this->Visible = false;
                }
                navigatorFormSettings->NavigatorFormLocation = this->Location;
                navigatorFormSettings->Save();
        }
private: System::Void NavigatorForm_Load(System::Object^  sender, System::EventArgs^
e)
        {
                try
                {
```

```
                        this->Location          =          navigatorFormSettings-
>NavigatorFormLocation;
                    }
                    catch (Exception^)
                    {

                    }
             }
protected: virtual property Boolean ShowWithoutActivation
                   {
                         virtual Boolean get() override
                         {
                               return true;
                         }
                   }
};
}
```

Nellipse.cpp

```
#include "StdAfx.h"
#include "NEllipse.h"
namespace NativeSource
{
namespace _NEllipse_ {
      void NEllipse::updateBoundRect()
      {
            const float a = S.Get(0,0);
            const float b = S.Get(1,1);
            const float cos_alpha = R.Get(0,0);
            const float sin_alpha = R.Get(1,0);
            const float x = sqrt( square(a*cos_alpha) + square(b*sin_alpha) );
            const float y = sqrt( square(a*sin_alpha) + square(b*cos_alpha) );
            /*выставить крайние точки и сдвинуть*/
            bound_rect.p1.x = Round(center.x - x);
            bound_rect.p1.y = Round(center.y - y);
            bound_rect.p2.x = Round(center.x + x);
            bound_rect.p2.y = Round(center.y + y);
            /*размеры*/
            UpdateSizeRect(bound_rect);
      }
      bool NEllipse::Contains( int x, int y, float& outDist )
      {
            NMatrix2x2f obrMatr = NMatrix2x2f(R*S);
            float determ;
            obrMatr = obrMatr.Obrat(determ);
            if(determ == 0)
            {
                  return false;
                  outDist = FLT_MAX;
            }
            NVector2f vec;
            vec.x = float(x-center.x);
            vec.y = float(y-center.y);
            vec = obrMatr*vec;
            float vecLen = ModulVector2(vec);
            if(vecLen <= 1.0f)
            {
                  outDist = vecLen;
                  return true;
```

```
		}
		else
		{
			outDist = vecLen;
			return false;
		}
	}
	void  NEllipse::Scale(  const  float  scale_koef,const  bool  isTranslate  /*=
false*/ )
	{
		S.Set(S.Get(0,0)*scale_koef,0,0,S.Get(1,1)*scale_koef);
		if(isTranslate == true)
		{
			center = center*scale_koef;
		}
		updateBoundRect();
	}
	void NEllipse::Scale( float s_00,float s_11 )
	{
		S.Set(S.Get(0,0)*s_00,0,0,S.Get(1,1)*s_11);
		updateBoundRect();
	}
	void NEllipse::Transform( float x, float y )
	{
		center.x += x;
		center.y += y;
		updateBoundRect();
	}
	void NEllipse::SetRotateMatrix( NMatrix2x2f matrix )
	{
		this->R = matrix;
		updateBoundRect();
	}
	void NEllipse::SetScaleMatrix( NMatrix2x2f matrix )
	{
		this->S = matrix;
		updateBoundRect();
	}
	void NEllipse::Rotate( float angle )
	{
		float cos_a = cos(angle);
		float sin_a = sin(angle);
		NMatrix2x2f new_R(cos_a,-sin_a,sin_a,cos_a);
		R = R*new_R;
		updateBoundRect();
	}
	float NEllipse::GetRad() const
	{
		return (S.Get(0,0) + S.Get(1,1))/2;
	}
	NativeSource::_NMathem_::NVector2f NEllipse::GetCenter() const
	{
		return center;
	}
	void NEllipse::SetCenter( float x,float y )
	{
		center.x = x;
		center.y = y;
		updateBoundRect();
	}
	float NEllipse::GetX()
```

```
	{
		return center.x;
	}
	float NEllipse::GetY()
	{
		return center.y;
	}
	NativeSource::_NMathem_::NBoundRecti NEllipse::GetBoundRect() const
	{
		return bound_rect;
	}
	NativeSource::_NMatrix_::NMatrix2x2f NEllipse::GetRoteMatrix() const
	{
		return R;
	}
	NativeSource::_NMatrix_::NMatrix2x2f NEllipse::GetScaleMatrix() const
	{
		return S;
	}
	bool NEllipse::Equals( NEllipse* val )
	{
		if(val == NULL)
		{
			return false;
		}
		else
		{
			return ((this->R == val->R)&(this->S == val->S)&(this->center ==
val->center));
		}
	}
	float NEllipse::GetArea()
	{
		return PI_F*abs(S.Get(0,0)*S.Get(1,1));
	}
	void NEllipse::CreateFromPoints( const int& x1, const int& y1, const int& x2,
const int& y2,const int& x3, const int& y3 ) /* расчет эллипса по трем точкам */
	{
		// расчитать центр эллипса
		float a = distance(x1,y1,x2,y2);
		float b = distance(x2,y2,x3,y3);
		float c = distance(x1,y1,x3,y3);
		if(a >= b && a >= c) // a
		{
			calculate(x1,y1,x2,y2,a,x3,y3);
		}
		else if(b >= a && b >= c) // b
		{
			calculate(x3,y3,x2,y2,b,x1,y1);
		}
		else // c
		{
			calculate(x1,y1,x3,y3,c,x2,y2);
		}
	}
	void NEllipse::CreateFromPoints2( const int& x1, const int& y1, const int& x2,
const int& y2,const int& x3, const int& y3 ) /* расчет эллипса по трем точкам */
	{
		// расчитать центр эллипса
		float a = distance(x1,y1,x2,y2);
		calculate(x1,y1,x2,y2,a,x3,y3);
```

```
	}
	void NEllipse::CreateFromPoints3( const int& x1, const int& y1, const int& x2, const int& y2,const int& x3, const int& y3 )
	{
		// проверяем какой прямоугольник
		float a = distance(x1,y1,x2,y2);
		float b = distance(x2,y2,x3,y3);
		float c = distance(x1,y1,x3,y3);
		float a_sq = square(a);
		float b_sq = square(b);
		float c_sq = square(c);
		if( (b_sq < a_sq + c_sq) && (c_sq < a_sq + b_sq) ) // острый
		{
			CreateFromPoints2(x1,y1,x2,y2,x3,y3);
		}
		else // прямоугольный или тупоугольный
		{
			CreateFromPoints(x1,y1,x2,y2,x3,y3);
		}
	}
	void NEllipse::calculate( const int& x1, const int& y1, const int& x2, const int& y2, const float& dist,const int& x3, const int& y3 ) /* первые две - большая полуось */
	{
		NVector2f vec3 = {x3,y3}; // третью точку в вектор
		center.x = float(x1+x2)/2; // центральная точка
		center.y = float(y1+y2)/2; // центральная точка
		float alpha = atan2(y1-center.y,x1-center.x); // -p/2 .. p/2 угол поворота a
		float cos_alpha = cos(alpha);
		float sin_alpha = sin(alpha);
		float a = dist/2;
		float a_sq = square(a);
		R.Set(cos_alpha,-sin_alpha,sin_alpha,cos_alpha);
		float determ;
		NMatrix2x2f R_Obrat = R.Obrat(determ);
		if(determ == 0)
		{
			R_Obrat = R;
		}
		// координаты третей точки повернутой обратно
		vec3 = vec3 - center;
		vec3 = R_Obrat*vec3;
		vec3 = square(vec3);
		// расчет второй полуоси
		float b = sqrt(vec3.y/(1.0f-vec3.x/a_sq));
		// заполнить матрицу масштабирования
		S.Set(a,0,0,b);
		//огранич прямоугольник
		updateBoundRect();
	}
}
}
```

NEllipse.h

```
	#pragma once
	#include "NMatrix2x2.h"
	#pragma unmanaged
	namespace NativeSource {
	namespace _NEllipse_ {
		using namespace NativeSource::_NMatrix_;
		using namespace NativeSource::_NMathem_;
		/*класс эллипса*/
```

```
	class NEllipse
	{
	public:
		NEllipse(void)
		{
			R.Set(1,0,0,1);
			S.Set(1,0,0,1);
			center.x = 0;
			center.y = 0;
			/*заполнить ограничивающий прямоугольник*/
			updateBoundRect();
		}
		NEllipse(const float rad,const float x,const float y)
		{
			R.Set(1,0,0,1);
			S.Set(rad,0,0,rad);
			center.x = x;
			center.y = y;
			/*заполнить ограничивающий прямоугольник*/
			updateBoundRect();
		}
		/* a - полуось
		   b - полуось
		   alpha - ориентация a
		   x,y - центр*/
		NEllipse(const float a,const float b,const float alpha,const float x,const
float y)
		{
			float cos_a = cos(alpha);
			float sin_a = sin(alpha);
			R.Set(cos_a,-sin_a,sin_a,cos_a);
			S.Set(a,0,0,b);
			center.x = x;
			center.y = y;
			/*заполнить ограничивающий прямоугольник*/
			updateBoundRect();
		}
		/*rr - ссылка на матрицу поворота
		  rs - ссылка на матрицу масштабирования
		  rc - ссылка на центр эллипса*/
		NEllipse(const NMatrix2x2f& rr,const NMatrix2x2f& rs,const NVector2f& rc)
		{
			R = rr;
			S = rs;
			center = rc;
			/*заполнить ограничивающий прямоугольник*/
			updateBoundRect();
		}
		~NEllipse(void)
		{
		}
	private:
		/*матрица поворота
		  cos(alpha) -sin(alpha)
		  sin(alpha)  cos(alpha)

		  alpha (-pi/2...pi/2)*/
		NMatrix2x2f R;
		/*матрица вращения
		  a 0
		  0 b*/
		NMatrix2x2f S;
		/*центр*/
		NVector2f center;
```

```
		/*прямоугольник*/
		NBoundRecti bound_rect;
		void updateBoundRect();
	public:
		/*определяет содержиться ли точка внутри эллипса
		 outDist - дистанция до центра от 0 до бесконечности
		  если меньше 1 значит содержитьсся внутри*/
		bool Contains(int x, int y, float& outDist);
		/*масштабировать
		isTranslate - флаг, определяет масштабирование центральной точки
		согласно масштабному коэффициенту*/
		void Scale(const float scale_koef,const bool isTranslate = false);
		/*перенести центр эллипса в указанную точку*/
		void Transform(float x, float y);
		void SetRotateMatrix(NMatrix2x2f matrix);
		void SetScaleMatrix(NMatrix2x2f matrix);
		/*повернуть эллипс на угол angle*/
		void Rotate(float angle);
		/*масштабировать по осям*/
		void Scale(float s_00,float s_11);
		float GetRad() const;
		NVector2f GetCenter() const;
		void SetCenter(float x,float y);
		float GetX();
		float GetY();
		NBoundRecti GetBoundRect() const;
		NMatrix2x2f GetRoteMatrix() const;
		NMatrix2x2f GetScaleMatrix() const;
		/*проверяет, эквивалентность эллипсов*/
		bool Equals(NEllipse* val);
		/*возвращает площадь эллипса*/
		float GetArea();
		/*заполнит эллипс по трем точкам в декартовой системе координат
		 большая полусось будет соответствовать двум наиболее удаленным точкам */
		void CreateFromPoints(const int& x1, const int& y1, const int& x2, const
int& y2,const int& x3, const int& y3); // расчет эллипса по трем точкам // пересчитывает
центр
		/*заполнит эллипс по трем точкам в декартовой системе координат
		  полусь определяется первыми двумя точками (x1 y1) (x2 y2)*/
		void CreateFromPoints2(const int& x1, const int& y1, const int& x2, const
int& y2,const int& x3, const int& y3); // расчет эллипса по трем точкам;
		/*заполнит эллипс по трем точкам в декартовой системе координат
		 1.  если  точки  образуют  остроугольный  треугольник  полуось  (малая  или
большая)
			определяется по первым двум точкам (x1 y1) (x2 y2)
		 2. если точки опразуют прямоугольный или тупоугольный треугольник большая
полусось будет
			соответствовать двум наиболее удаленным точкам*/
		void CreateFromPoints3(const int& x1, const int& y1, const int& x2, const
int& y2,const int& x3, const int& y3);
	private:
		/*рассчитывает эллипс по трем точкам в декартовой системе координат
		 (x1 y1) (x2 y2) - точки лежащие на полуоси (малой или большой)
		  dist - евклидово растояние между (x1 y1) (x2 y2)
		  (x3 y3) - третья координата, через которую рассчитывается еллипс*/
		void calculate(const int& x1, const int& y1, const int& x2, const int& y2,
const float& dist,const int& x3, const int& y3); // первые две - большая полуось;
	};
	}
	}
	#pragma managed
```

OneImage.h

```
#pragma once
#include "OpenImageDialog.h"
#include "PropertySorter.h"
#include "ImageMetadataCollection.h"
#include "IRequestPerformer.h"
#include "IResultBuilder.h"
#include "OneImageResultData.h"
#include "AsyncOperation.h"
#include "ModelRequest.h"
#include "IComand.h"
namespace Forest {
	using namespace System;
	using namespace System::Design;
	using namespace System::ComponentModel;
	using namespace System::ComponentModel::Design;
	using namespace System::Collections;
	using namespace System::Collections::Generic;
	using namespace System::Windows::Forms;
	using namespace System::Windows::Forms::Design;
	using namespace System::Data;
	using namespace System::Drawing;
	using namespace System::Drawing::Imaging;
	using namespace System::Drawing::Design;
	using namespace System::Drawing::Drawing2D;
	using namespace System::Reflection;
	using namespace System::IO;
	using namespace System::Drawing::Imaging;
	using namespace AppRequest;
	using namespace AppRequest::ModelRequest;
	using namespace AppComands;
	using namespace ReportingLib;
/*команда "загрузить изображение"*/
ref class LoadImageComand: public IComand
{
public:
	LoadImageComand(String^ fName)
	{

		fileName = fName;
		comandInfo = "Выполняется загрузка изображения";
	}
protected:
	~LoadImageComand()
	{
		fileName = nullptr;
		comandInfo = nullptr;
		image = nullptr;
		this->!LoadImageComand();
	}
	!LoadImageComand()
	{

	}
private:
	/*изображение*/
	Image^ image;
	/*инфпрмация о команде*/
	String^ comandInfo;
```

```
	/*имя открываемого файла*/
	String^ fileName;
#pragma region реализация интерфейса IComand
public:
	/*выполнить*/
	virtual void execute()
	{
		this->image = Image::FromFile(fileName);
	}
	/*Информация о команде*/
	virtual property String^ ComandInfo
	{
		String^ get()
		{
			return comandInfo;
		}
	}
#pragma endregion интерфейс IComand
public:
	/*возвращаемый результат*/
	property Image^ Image_
	{
		Image^ get()
		{
			return image;
		}
	}
};
/*изображение с которым работаем*/
[TypeConverter(PropertySorter::typeid)] /// указываем, что класс должен сортироваться
ref class OneImage: public IRequestPerformer, public IResultBuilder,public
IReportData
{
public:
	OneImage(Int32 numb,OneImageResultData^ resData):number(numb),
fileName(emptyFileName), oneImageResultData(resData)
	{
		displayName = "Изображение " + NumberStr;
		/*создаем себе копию асинхронной операции*/
		asyncOperation = AsyncOperation::InstanceOperation;
		asyncOperation->StartAsinc += gcnew
AsincNotificationEventHandler(this,&OneImage::asyncOperation_StartAsinc);
		asyncOperation->StopAsinc += gcnew
AsincNotificationEventHandler(this,&OneImage::asyncOperation_StopAsinc);
	}
protected:
	~OneImage()
	{
		asyncOperation->StartAsinc -= gcnew
AsincNotificationEventHandler(this,&OneImage::asyncOperation_StartAsinc);
		asyncOperation->StopAsinc -= gcnew
AsincNotificationEventHandler(this,&OneImage::asyncOperation_StopAsinc);
		asyncOperation = nullptr;
		if(lastExecutedComand != nullptr)
		{
			delete lastExecutedComand;
			lastExecutedComand = nullptr;
		}
		if(metaDataCollection != nullptr)
		{
			delete metaDataCollection;
```

```
                metaDataCollection = nullptr;
            }
            image = nullptr;
            oneImageResultData = nullptr;
            this->!OneImage();
        }
        !OneImage()
        {
        }
private:
#pragma region приватные поля
        static initonly String^ emptyFileName = "< Открыть... >";
        /*изображение*/
        Image^ image;
        /*полное имя файла*/
        String^ fileName;
        /*имя*/
        String^ displayName;
        /*метаданные*/
        ImageMetadataCollection^ metaDataCollection;
        /*последняя выполненная команда*/
        IComand^ lastExecutedComand;
        /*Асинхронная операция*/
        AsyncOperation^ asyncOperation;
        /*данные, которые заполняет*/
        OneImageResultData^ oneImageResultData;
        /*порядковый номер изображения*/
        Int32 number;
#pragma endregion приватные поля
public:
        /*изображение с которым работаем*/
        [BrowsableAttribute(false)]
        [PropertyOrder(10)]
        property Image^ Image_
        {
            Image^ get()
            {
                return image;
            }
            void set(Image^ val)
            {
                image = val;
                if(metaDataCollection != nullptr)
                {
                    delete metaDataCollection;
                    metaDataCollection = nullptr;
                }
                metaDataCollection = gcnew ImageMetadataCollection(image);
                /*сообщение изменилось свойство класса*/
                this->PropertyChanged(this,nullptr);
            }
        }
        /*путь до файла*/
        [ReadOnlyAttribute(false)]
        [BrowsableAttribute(true)]
        [DisplayName("Имя")]
        [Description("Задает путь к изображению")]
        [PropertyOrder(20)]
        [Editor(ImageNameEditor::typeid                                                    ,
System::Drawing::Design::UITypeEditor::typeid)]
        property String^ FileName
```

```
{
	String^ get()
	{
		return fileName;
	}
	void set(String^ val)
	{
		if((fileName = val) != nullptr)
		{
			/*открываем файл асинхронно*/
			/* удаляем предвдущую команду */
			if(lastExecutedComand != nullptr)
			{
				delete lastExecutedComand;
				lastExecutedComand = nullptr;
			}
			/*создаем новую*/
			lastExecutedComand = gcnew LoadImageComand(fileName);
			/*запускаем асинхронно*/
			asyncOperation->Execute(lastExecutedComand);
		}
	}
}
/*имя*/
[BrowsableAttribute(false)]
[PropertyOrder(30)]
property String^ DisplayName
{
	String^ get()
	{
		return displayName;
	}
	void set(String^ val)
	{
		displayName = val;
	}
}
/*фомат*/
[ReadOnlyAttribute(false)]
[BrowsableAttribute(true)]
[DisplayName("Формат")]
[Description("Формат изображения")]
[PropertyOrder(40)]
property ImageFormat^ RawFormat
{
	ImageFormat^ get()
	{
		return image != nullptr ? image->RawFormat : nullptr;
	}
}
/*высота*/
[ReadOnlyAttribute(false)]
[BrowsableAttribute(true)]
[DisplayName("Высота")]
[Description("Высота изображения в пикселях")]
[PropertyOrder(50)]
property Int32 Height
{
	Int32 get()
	{
		return image != nullptr ? image->Height : 0;
```

```
	}
}
/*ширина*/
[ReadOnlyAttribute(false)]
[BrowsableAttribute(true)]
[DisplayName("Ширина")]
[Description("Ширина изображения в пикселях")]
[PropertyOrder(60)]
property Int32 Width
{
	Int32 get()
	{
		return image != nullptr ? image->Width : 0;
	}
}
/*вериткальное разрешение*/
[ReadOnlyAttribute(false)]
[BrowsableAttribute(true)]
[DisplayName("Вертикальное разрешение")]
[Description("Вертикальное разрешение изображения в пикселях на дюйм")]
[PropertyOrder(70)]
property Single VerticalResolution
{
	Single get()
	{
		return image != nullptr ? image->VerticalResolution : 0.0f;
	}
}
/*горизонтальное разрешение*/
[ReadOnlyAttribute(false)]
[BrowsableAttribute(true)]
[DisplayName("Горизонтальное разрешение")]
[Description("Горизонтальное разрешение изображения в пикселях на дюйм")]
[PropertyOrder(80)]
property Single HorizontalResolution
{
	Single get()
	{
		return image != nullptr ? image->HorizontalResolution : 0.0f;
	}
}
/*метаданные*/
[ReadOnlyAttribute(true)]
[BrowsableAttribute(true)]
[DisplayName("Метаданные")]
[Description("Метаданные изображения")]
[PropertyOrder(90)]
[Editor(NoneEditor::typeid, UITypeEditor::typeid)]
property ImageMetadataCollection^ MetaDataCollection
{
	ImageMetadataCollection^ get()
	{
		return metaDataCollection;
	}
}
/*порядковый номер*/
[BrowsableAttribute(false)]
property Int32 Number
{
	Int32 get()
	{
```

```
                return number;
            }
        }
        /*порядковый номер (строковое представление)*/
        [BrowsableAttribute(false)]
        property String^ NumberStr
        {
            String^ get()
            {
                return "№ "+ number.ToString();
            }
        }
        /*изменилилось какое-нибудь свойство*/
        event System::EventHandler^ PropertyChanged;
        /*преобразуем изображение в */
        virtual String^ ToString() override
        {
            return this->image == nullptr ? (L"(Отсутствует)") : (L"\u2611");
        }
        /*открыть изображение*/
private: void openImage()
        {
            if(OpenImageDialog::GetOpenImageDialog()->ShowDialog()                ==
System::Windows::Forms::DialogResult::OK)
            {
                this->FileName = OpenImageDialog::GetOpenImageDialog()->FileName;
            }
        }
#pragma region реализация интерфейса IResultBuilder
public:
        virtual System::Void Build(IResultData^ iResultData_)
        {
            /*данные сбрасываются, когда открывается новое изображение
             или изображения не совпадают*/
            iResultData_->ResetData(); // сбросим данные
            array<String^>^ names = gcnew array<String^>(1){this->displayName};
            array<Image^>^ images = gcnew array<Image^>(1){this->Image_};
            iResultData_->SourceData = gcnew SourceImages(names,images,0);
        }
#pragma endregion реализация интерфейса IResultBuilder
#pragma region реализация интерфейса IRequestPerformer
public:
       /*запрос на выполнение*/
       virtual void RequestToExecute(IModelRequest^ modelRequest)
       {
            switch ( modelRequest->ID )
            {
                /*запросы к изображению*/
            case MODEL_REQUEST_ID::OPEN_IMAGE: // открыть изображение
                this->openImage();
                break;
            case MODEL_REQUEST_ID::UPDATE_RESULT_DATA: // обновить данные
                {
                    if(lastExecutedComand != nullptr)
                    {
                        delete lastExecutedComand;
                        lastExecutedComand = nullptr;
                    }
                    lastExecutedComand                =                gcnew
FillResultDataComand(this,oneImageResultData,nullptr);
                    lastExecutedComand->execute();
```

```
                        this->RequestExecuted(this,nullptr);
                    }
                    break;
                default:
                    MessageBox::Show(this->ToString() + ": Не знаю, как выполнить
запрос " + modelRequest->ToString());

                }
            }
            /*событие "Запрос выполнен"*/
            virtual event System::EventHandler^ RequestExecuted;
#pragma endregion реализация интерфейса IRequestPerformer
#pragma region обработчики событий asyncOperation
private:
        System::Void asyncOperation_StartAsinc(Object^ sender,Object^ args)
        {
                //не обрабатываем
        }
        System::Void asyncOperation_StopAsinc(Object^ sender,Object^ args)
        {
                AsyncOperation^ ao = dynamic_cast<AsyncOperation^>(sender);
                /*получить результат*/
                if(ao->ExecutableComand == this->lastExecutedComand) // работаем только
со своей командой
                {
                        /*выгружаем себе изображение из команды*/
                        this->Image_                                                    =
dynamic_cast<LoadImageComand^>(lastExecutedComand)->Image_;
                        /*запускаем команду на заполнение данных*/
                        this->RequestToExecute(gcnew MrUpdateResultData);
                }
        }
#pragma endregion обработчики событий asyncOperation
#pragma region реализация интерфейса IReportData
public:
        /*поддерживаемые секции (перечисление)*/
        [BrowsableAttribute(false)]
        virtual property ReportSections Sections
        {
                ReportSections get()
                {
                        return ReportSections::IMAGE_PARAM;
                }
        }
        /*секция изображение*/
        [BrowsableAttribute(false)]
        virtual property List<ReportImage^>^ ImgSection
        {
                List<ReportImage^>^ get()
                {
                        return createImgSection();
                }
        }
        /*секция параметров*/
        [BrowsableAttribute(false)]
        virtual property List<ReportParam^>^ ParamSection
        {
                List<ReportParam^>^ get()
                {
                        return createParamSection();
                }
```

```
	}
	[BrowsableAttribute(false)]
	virtual property List<ReportHist^>^ HistSection
	{
		List<ReportHist^>^ get()
		{
			return nullptr;
		}
	}
private:
	/*создать секцию изображений*/
	List<ReportImage^>^ createImgSection()
	{
		List<ReportImage^>^ imgSection = gcnew List<ReportImage^>(1);
		if(this->image != nullptr)
		{
			System::IO::MemoryStream^ memoryStream = gcnew
System::IO::MemoryStream();
			try
			{
				String^ mimeStr = "image/png"; // мime
				/*изображение в поток*/
				Image^ img = gcnew Bitmap(this-
>image,Global::FitSizeOn(Drawing::Size(image->Width,image-
>Height),Drawing::Size(768,576)));
				img->Save(memoryStream,ImageFormat::Png);
				memoryStream->Seek(0, System::IO::SeekOrigin::Begin);
				/*из потока в строку*/
				String^ sourceStr = Convert::ToBase64String(memoryStream-
>GetBuffer()); //ресурсы
				/*создание ReportImage*/
				ReportImage^ reportImg = gcnew
ReportImage(displayName,sourceStr,mimeStr,displayName);
				imgSection->Add(reportImg);
			}
			finally
			{
				memoryStream->Close();
				delete memoryStream;
				memoryStream = nullptr;
			}
		}
		return imgSection;
	}
	/*создать секцию параметровв*/
	List<ReportParam^>^ createParamSection()
	{
		List<ReportParam^>^ paramSection = gcnew List<ReportParam^>();
		fillSimpleDataParam(paramSection);
		fillMetaDataParam(paramSection);
		return paramSection;
	}
	/*заполнить простыми данными*/
	System::Void fillSimpleDataParam(List<ReportParam^>^ paramSection_)
	{

		if(this->image != nullptr)
		{
			paramSection_->Add(gcnew ReportParam("Изображение","Имя
изображения",this->FileName,""));
```

```
                  paramSection_->Add(gcnew         ReportParam("Изображение","Формат
изображения",(TypeDescriptor::GetConverter(ImageFormat::typeid))-
>ConvertToString(this->RawFormat),""));
                  paramSection_->Add(gcnew        ReportParam("Изображение","Высота,
пикс",this->Height.ToString(),""));
                  paramSection_->Add(gcnew        ReportParam("Изображение","Ширина,
пикс.",this->Width.ToString(),""));
            }
      }
      /*заполнить данными из метаданных*/
      System::Void fillMetaDataParam(List<ReportParam^>^ paramSection_)
      {
            if(this->image != nullptr)
            {
                  if(this->metaDataCollection != nullptr)
                  {
                        for(int i = 0;i<metaDataCollection->Count;++i)
                        {
                              paramSection_->Add(gcnew ReportParam("Изображение",

      metaDataCollection->Item[i]->Name,

      metaDataCollection->Item[i]->Value,

      ""));
                        }
                  }
            }
      }
#pragma endregion реализация интерфейса IReportData



};

}
```

OneImageCubaturnic.h

```
#pragma once
#include "NStackLogsDetect.h"
#include "ModelRequest.h"
#include "IComand.h"
#include "AsyncOperation.h"
#include "IRequestPerformer.h"
#include "IResultBuilder.h"
#include "OneImageResultData.h"
#include "OneImageCubaturnicMethods.h"
namespace Forest
{
      using namespace System;
      using namespace System::ComponentModel;
      using namespace System::Collections;
      using namespace System::Collections::Generic;
      using namespace System::Windows::Forms;
      using namespace System::Data;
      using namespace System::Drawing;
      using namespace System::Drawing::Drawing2D;
      using namespace System::Reflection;
      using namespace ImageConverting;
```

```
	using namespace NativeSource;
	using namespace NativeSource::_NStackLogsDetect_;
	using namespace NativeSource::_NResults_;
	using namespace NativeSource::_NEllipse_;
	using namespace AppRequest;
	using namespace AppRequest::ModelRequest;
	using namespace AppComands;
	using namespace Global;
	using namespace CubaturnicLib;
#pragma region преобразователь типов для выбора метода расчета
	ref class Gost2708MethodsType : ExpandableObjectConverter
	{
	public:	virtual	bool	GetStandardValuesSupported(ITypeDescriptorContext^
context ) override;
	public:	virtual	bool	GetStandardValuesExclusive(ITypeDescriptorContext^
context ) override;
	public:			virtual			StandardValuesCollection^
GetStandardValues(ITypeDescriptorContext^ context ) override;
	public: virtual bool CanConvertFrom(ITypeDescriptorContext^ context, Type^
destinationType) override;
	public:		virtual		Object^		ConvertFrom(ITypeDescriptorContext^
context,System::Globalization::CultureInfo^ culture, Object^ value) override;
	};
#pragma endregion преобразователь типов для выбора метода расчета
/* класс кубатурника
   */
[TypeConverter(PropertySorter::typeid)]
ref class OneImageCubaturnic: public IRequestPerformer, public IResultBuilder, public
IReportData
{
public:
	OneImageCubaturnic(OneImageResultData^ resData): oneImageResultData(resData)
	{
		/*создадим себе ссылку на асинхронную операцию*/
		asyncOperation = AsyncOperation::InstanceOperation;
		asyncOperation->StartAsinc			+=			gcnew
AsincNotificationEventHandler(this,&OneImageCubaturnic::asyncOperation_StartAsinc);
		asyncOperation->StopAsinc			+=			gcnew
AsincNotificationEventHandler(this,&OneImageCubaturnic::asyncOperation_StopAsinc);
		/*создаем методы расчета*/
		methods = gcnew List<OneImageCubMethodBase^>();
		methods->Add(gcnew OneImageCubTablesGOST2708());
		methods->Add(gcnew OneImageCubTopDiameterMeanRise());
		/*подписываемся на изменение свойств методов*/
		for(int i = 0;i<methods->Count;++i)
		{
			methods[i]->PropertyChanged		+=		gcnew
System::EventHandler(this,&OneImageCubaturnic::methods_PropertyChanged);
		}
		selectedMethod = methods[0];
		crownPercent = 50;
		cb_rand = gcnew Random(134);
	}
protected:
	~OneImageCubaturnic()
	{
		asyncOperation->StartAsinc			-=			gcnew
AsincNotificationEventHandler(this,&OneImageCubaturnic::asyncOperation_StartAsinc);
		asyncOperation->StopAsinc			-=			gcnew
AsincNotificationEventHandler(this,&OneImageCubaturnic::asyncOperation_StopAsinc);
		asyncOperation = nullptr;
```

```
            oneImageResultData = nullptr;
            if(lastExecutedComand != nullptr)
            {
                  delete lastExecutedComand;
                  lastExecutedComand = nullptr;
            }
            selectedMethod = nullptr;
            for each(OneImageCubMethodBase^ met in methods)
            {
                  met->PropertyChanged                    -=                    gcnew
System::EventHandler(this,&OneImageCubaturnic::methods_PropertyChanged);
                  met = nullptr;
            }
            methods->Clear();
            methods = nullptr;
            this->!OneImageCubaturnic();
      }
      !OneImageCubaturnic()
      {
      }
#pragma region приватные поля
private:
      /*данные, которые заполняет калибровщик*/
      OneImageResultData^ oneImageResultData;
      /*Асинхронная операция*/
      AsyncOperation^ asyncOperation;
      /*последняя выполненная команда*/
      IComand^ lastExecutedComand;
      Int32 crownPercent;  // процент вершинных бревен
      Single volumeCorrectionCoeff; // поправочный коэффициент на объем
      OneImageCubMethodBase^ selectedMethod; // выбранный метод
      List<OneImageCubMethodBase^>^ methods; // список доступных методов
      Random^ cb_rand; // случайное число, определяющее комель или вершина
#pragma endregion приватные поля
#pragma region видимые свойства
public:
      [ReadOnlyAttribute(false)]
      [BrowsableAttribute(true)]
      [DisplayName("Процент вершинных бревен")]
      [Description("Определяет процент вершинных бревен на изображеннии")]
      [Editor(PercentDropDownEditor::typeid, UITypeEditor::typeid)]
      [PropertyOrder(10)]
      property Int32 CrownPercent
      {
            Int32 get()
            {
                  return crownPercent;
            }
            void set(Int32 val)
            {
                  crownPercent = val;
                  this->methods_PropertyChanged(this,nullptr);
            }
      }
      [ReadOnlyAttribute(false)]
      [BrowsableAttribute(true)]
      [DisplayName("Поправочный коэффициент на объем коры")]
      [Description("Поправочный коэффициент на объем коры")]
      //[Editor(PercentDropDownEditor::typeid, UITypeEditor::typeid)]
      [PropertyOrder(20)]
      property Single VolumeCorrectionCoeff
```

```
	{
		Single get()
		{
			return volumeCorrectionCoeff;
		}
		void set(Single val)
		{
			volumeCorrectionCoeff = val;
			this->methods_PropertyChanged(this,nullptr);
		}
	}
	[ReadOnlyAttribute(false)]
	[BrowsableAttribute(true)]
	[DisplayName("Метод расчета")]
	[Description("Метод расчета")]
	[TypeConverter(Gost2708MethodsType::typeid)]
	[PropertyOrder(30)]
	property OneImageCubMethodBase^ SelectedMethod
	{
		OneImageCubMethodBase^ get()
		{
			return selectedMethod;
		}
		void set(OneImageCubMethodBase^ value)
		{
			selectedMethod = value;
			this->methods_PropertyChanged(this,nullptr);
		}
	}
	[BrowsableAttribute(false)]
	property List<OneImageCubMethodBase^>^ Methods
	{
		List<OneImageCubMethodBase^>^ get()
		{
			return methods;
		}
	}
#pragma endregion видимые свойства
#pragma region реализация интерфейса IRequestPerformer
public:
	/*запрос на выполнение*/
	virtual void RequestToExecute(IModelRequest^ modelRequest)
	{
		/*
		1. определяем, что за запрос
		2. упаковываем запрос в команду,
		точнее берем данные из запроса и подсовываем в
		команду
		3. запускаем команду асинхронно (в потоке)
		*/
		switch ( modelRequest->ID )
		{
		case MODEL_REQUEST_ID::UPDATE_RESULT_DATA: // добавить объект
			/* создаем команду и выполняем ее асинхронно */
			if(asyncOperation->IsBusy == false)
			{
				/* удаляем предвдущую команду */
				if(lastExecutedComand != nullptr)
				{
					delete lastExecutedComand;
					lastExecutedComand = nullptr;
```

```
                        }
                        lastExecutedComand                =                gcnew
FillResultDataComand(this,oneImageResultData,"");
                        lastExecutedComand->execute();
                        this->RequestExecuted(this,nullptr);
                  }
                  break;
            default:
                  MessageBox::Show( this->ToString() + ": Не знаю, как выполнить
запрос " + modelRequest->ToString());
            }
      }
      /*событие "Запрос выполнен"*/
      virtual event System::EventHandler^ RequestExecuted;
#pragma endregion реализация интерфейса IRequestPerformer
      /*изменилилось какое-нибудь свойство*/
      event System::EventHandler^ PropertyChanged;
#pragma region обработчики событий asyncOperation
private:
      System::Void asyncOperation_StartAsinc(Object^ sender,Object^ args)
      {
            //не обрабатываем
      }
      System::Void asyncOperation_StopAsinc(Object^ sender,Object^ args)
      {
      }
#pragma endregion обработчики событий asyncOperation
#pragma region реализация интерфейса IResultBuilder
public:
     virtual System::Void Build(IResultData^ iResultData_)
     {
            // распоковать данные
            CutLogsCollection^                        items_                        =
dynamic_cast<CutLogsCollection^>(iResultData_->CollectionData);
            /*сбросить данные кубатурника*/
            items_->ResetCubaturnicData();
            if(items_->IsCalibrated == true)
            {
                /*присвоить ориентацию согласно crownPersent*/

     if(fillOrientation(dynamic_cast<OneImageResultData^>(iResultData_))/*fillOrien
tation_Random(dynamic_cast<OneImageResultData^>(iResultData_))*/ == true)
                {
                      /*вызвать выбранный пользователем метод*/
                      SelectedMethod->Build(iResultData_);
                      /*запоняем сумарный результат если для коллекции выставлен
соответствующий флаг*/
                      if(items_->IsCalculated == true)
                      {
                            Single cub = 0.0f;
                            for(int i = 0; i < items_->Count;++i )
                            {
                                  cub += items_->Item[i]->Cubature;
                            }
                            items_->StackCubature                                 =
CubaturnicLib::Converter::VolumeToString(cub);
                      }
                      else
                      {
                            /*у нерасчитанных выставить размеры по умолчанию*/
```

```
                                List<CutLogsItem^>^     uc_items_     =     items_-
>GetUncalculatedItems();
                                for each(CutLogsItem^ uc_item_ in uc_items_)
                                {
                                      Single uc_diam = uc_item_->MeanDiameterMm/10;
                                      uc_diam                                    =
CubaturnicLib::g_52117_2003_Rounder::CmDiameter(uc_diam);
                                      uc_item_->Lenght       =       SelectedMethod-
>TruthSelectedLength;
                                }
                          }}
            }}
#pragma endregion реализация интерфейса IResultBuilder
public: virtual String^ ToString() override
            {
                  return selectedMethod != nullptr ? selectedMethod->ToString():"";
            }
            /*выставить ориентацию в штабеле*/
private: Boolean fillOrientation(OneImageResultData^ obj)
{
            /*получить коллекцию с неопределенной ориентацией*/
          CutLogsCollection^    items_    =    dynamic_cast<CutLogsCollection^>(obj-
>CollectionData);
            List<CutLogsItem^>^ indefItems = items_->GetIndefOrientationItems();
            /*сортируем по возрастанию диаметра*/
            try
            {
                  indefItems->Sort(gcnew LessByMeanDiameterMmComparer);
            }
            catch (Exception^ e)
            {
                  MessageBox::Show(e->Message);
            }
            /*расчитываем limit_count*/
            int         limit_count         =         (Int32)Math::Round((indefItems-
>Count*crownPercent)/100.0f);
            for(int i = 0;i < indefItems->Count;++i)
            {
                  indefItems[i]->SetOrientation(       (       i<limit_count       ?
ORIENTATION_LOG::CROWN : ORIENTATION_LOG::BUTT ) ,false);
            }
            return items_->GetIndefOrientationItems()->Count == 0;
}
             /*заполнить данные "комель-вершина" случайно*/
private: Boolean fillOrientation_Random(OneImageResultData^ obj)
             {
                   /*получить коллекцию с неопределенной ориентацией*/
                   CutLogsCollection^                    items_                    =
dynamic_cast<CutLogsCollection^>(obj->CollectionData);
                   List<CutLogsItem^>^         indefItems         =         items_-
>GetIndefOrientationItems();
                   /*сортируем по возрастанию диаметра*/
                   try
                   {
                         indefItems->Sort(gcnew LessByMeanDiameterMmComparer);
                   }
                   catch (Exception^ e)
                   {
                         MessageBox::Show(e->Message);
                   }
                   /*рандомно присвоить комель/вершина*/
```

```
                    for(int i = 0;i < indefItems->Count;++i)
                    {
                          Int32 rand_val = cb_rand->Next(0,2);
                          if(rand_val == 0)
                               indefItems[i]->SetOrientation(ORIENTATION_LOG::CROWN
,false);
                          else if(rand_val == 1)
                                indefItems[i]->SetOrientation(ORIENTATION_LOG::BUTT
,false);
                          else
                                MessageBox::Show("Ошибка    присвоения    рандомного
числа");
                    }
                    return items_->GetIndefOrientationItems()->Count == 0;
              }
private: void methods_PropertyChanged(Object^ sender, System::EventArgs^ e)
              {
                    /*отправить сообщение выше*/
                    this->PropertyChanged(this,nullptr);
                    /*запустить пересчкт*/
                    this->RequestToExecute(gcnew MrUpdateResultData);
              }
#pragma region реализация интерфейса IReportData
public:
      /*поддерживаемые секции (перечисление)*/
      [BrowsableAttribute(false)]
      virtual property ReportSections Sections
      {
            ReportSections get()
            {
                  return         selectedMethod         ==         nullptr         ?
(ReportSections::PARAM):(ReportSections::PARAM | selectedMethod->Sections);
            }}
      /*секция изображение*/
      [BrowsableAttribute(false)]
      virtual property List<ReportImage^>^ ImgSection
      {
            List<ReportImage^>^ get()
            {
                  if(selectedMethod != nullptr)
                  {
                        if((selectedMethod->Sections  &  ReportSections::IMAGE)  ==
ReportSections::IMAGE)
                        {
                              /*вернуть изображения секции*/
                              return selectedMethod->ImgSection;
                        }
                  }
                  return nullptr;
            }}
      /*секция параметров*/
      [BrowsableAttribute(false)]
      virtual property List<ReportParam^>^ ParamSection
      {
            List<ReportParam^>^ get()
            {
                  return createParamSection();
            }}
      [BrowsableAttribute(false)]
      virtual property List<ReportHist^>^ HistSection
      {
```

```
            List<ReportHist^>^ get()
            {
                  if(selectedMethod != nullptr)
                  {
                        if((selectedMethod->Sections   &   ReportSections::HIST)   ==
ReportSections::HIST)
                        {
                              /*вернуть гистограмму секции*/
                              return selectedMethod->HistSection;
                        }
                  }
                  return nullptr;
            }
      }
private:
      List<ReportParam^>^ createParamSection()
      {
            List<ReportParam^>^ paramSection = gcnew List<ReportParam^>;
            if(selectedMethod != nullptr)
            {
                  paramSection->Add(gcnew            ReportParam("Кубатурник","Метод
расчета",selectedMethod->ToString(),""));
                  paramSection->Add(gcnew      ReportParam("Кубатурник","Поправочный
коэффициент на объем коры",volumeCorrectionCoeff.ToString(),""));
                  paramSection->Add(gcnew          ReportParam("Кубатурник","Процент
вершинных бреввен",crownPercent.ToString()+" %",""));
                  if((selectedMethod->Sections    &    ReportSections::PARAM)    ==
ReportSections::PARAM)
                  {
                        paramSection->AddRange(selectedMethod->ParamSection);
                  }}
            return paramSection;
      }
#pragma endregion реализация интерфейса IReportData
};}
```

OneImageModel.h

```
#pragma once
#include "IModelImageStrategy.h"
#include "OneImage.h"
#include "OneImageDetector.h"
#include "OneImageCalibrator.h"
#include "OneImageCutLogsCollection.h"
#include "OneImageResultData.h"
#include "PropertyUIEditor.h"
#include "GraphicSettings.h"
#include "OneImageCubaturnic.h"
namespace Forest
{
      using namespace System;
      using namespace System::Design;
      using namespace System::ComponentModel;
      using namespace System::ComponentModel::Design;
      using namespace System::Collections;
      using namespace System::Collections::Generic;
      using namespace System::Windows::Forms;
      using namespace System::Windows::Forms::Design;
      using namespace System::Data;
      using namespace System::Drawing;
```

```
	using namespace System::Drawing::Design;
	using namespace System::Drawing::Drawing2D;
	using namespace System::Reflection;
	using namespace Forest::SavedSettings;
	using namespace ReportingLib;
#pragma region преобразователь типов для выбора метода калибровки
	ref class OneImageCalibrationMethodsType : ExpandableObjectConverter
	{
	protected:
		!OneImageCalibrationMethodsType()
		{
		}
	public:	virtual	bool	GetStandardValuesSupported(ITypeDescriptorContext^
context ) override;
	public:	virtual	bool	GetStandardValuesExclusive(ITypeDescriptorContext^
context ) override;
	public:			virtual			StandardValuesCollection^
GetStandardValues(ITypeDescriptorContext^ context ) override;
	public: virtual bool CanConvertFrom(ITypeDescriptorContext^ context, Type^
destinationType) override;
	public: virtual bool GetPropertiesSupported(ITypeDescriptorContext^ context)
override;
	public:		virtual		Object^		ConvertFrom(ITypeDescriptorContext^
context,System::Globalization::CultureInfo^ culture, Object^ value) override;
	};
#pragma endregion преобразователь типов для выбора мктода калибровки
	/*вся модель для работы с одним изображением*/
ref class OneImageModel: IModelImagesStrategy
{
public:
	OneImageModel(void)
	{
		/*результирующие данные, которые заполняем и отправляем на форму*/
		oneImageResultData = gcnew OneImageResultData();
		/*изображение*/
		oneImage = gcnew OneImage(1,oneImageResultData);
		oneImage->RequestExecuted			+=			gcnew
System::EventHandler(this,&OneImageModel::oneImage_RequestExecuted);
		oneImage->PropertyChanged			+=			gcnew
System::EventHandler(this,&OneImageModel::oneImageModel_PropertyChanged);
		/*менеджер отчетов*/
		createReportManager();
		/*кубатурник*/
		createOneImageCubaturnic();
		/*калибровщик*/
		//весь список
		createOneImageCalibrationMethods();
		//конкрктного конкрктного
		createOneImageCalibrator(IMG_CALIBRATION_METHODS::KnownDiameter/*эту
величину взять из настроек*/);
	}
	OneImageModel(Int32 num_img)
	{
		/*результирующие данные, которые заполняем и отправляем на форму*/
		oneImageResultData = gcnew OneImageResultData();
		/*изображение*/
		oneImage = gcnew OneImage(num_img,oneImageResultData);
		oneImage->RequestExecuted			+=			gcnew
System::EventHandler(this,&OneImageModel::oneImage_RequestExecuted);
		oneImage->PropertyChanged			+=			gcnew
System::EventHandler(this,&OneImageModel::oneImageModel_PropertyChanged);
```

```
			/*менеджер отчетов*/
			createReportManager();
			/*кубатурник*/
			createOneImageCubaturnic();
			/*калибровщик*/
			//весь список
			createOneImageCalibrationMethods();
			//конкрктного конкрктного
			createOneImageCalibrator(IMG_CALIBRATION_METHODS::KnownDiameter/*эту
величину взять из настроек*/);
		}
protected:
		virtual ~OneImageModel()
		{
			clearReportManager();
			clearOneImageCubaturnic();
			oneImageCalibrator = nullptr;
			this->clearOneImageCalibrationMethods();
			clearOneImage();
			clearOneImageDetector();

			/*!!!!!!!!!!!!!!!!!!!!!!*/
			if(oneImageResultData != nullptr)
			{
				oneImageResultData->ResetData();
				oneImageResultData = nullptr;
			}
			image = nullptr;
			this->!OneImageModel();
		}
		!OneImageModel()
		{
		}
#pragma region поля
protected: // видимы в дочерних классах
		Image^ image;
		/*изображение*/
		OneImage^ oneImage;
		/*объект детектирования объектов*/
		OneImageDetector^ oneImageDetector;
		/*объект калибровщика*/
		OneImageBaseCalibrator^ oneImageCalibrator;
		/*объект расчета 3d модели*/
		/*объект кубатурника*/
		OneImageCubaturnic^ oneImageCubaturnic;
		/*объект результирующие данные*/
		OneImageResultData^ oneImageResultData;
		/*стратегия*/
		static		initonly		MODEL_IMAGE_STRATEGIES		modelStrategy		=
MODEL_IMAGE_STRATEGIES::OneImg;
		/*массив доступных методов калибровки*/
		Dictionary<IMG_CALIBRATION_METHODS,OneImageBaseCalibrator^>^
oneImageCalibrationMethods;
		/*менеджер отчетов*/
		ReportingLib::ReportManager^ reportManager;
#pragma endregion поля
#pragma region обработчики событий OneImage
		/*изображение выполнело запрос*/
private: System::Void oneImage_RequestExecuted(Object^ sender,System::EventArgs^ e)
		{
			/*создаем объект детектора  и калибровщика*/
```

```
            OneImage^ oi = dynamic_cast<OneImage^>(sender);
            if(oi->Image_ != nullptr)
            {
                  if(this->image != oi->Image_)
                  {
                        /*инициируем новое изображение*/
                        this->image = oi->Image_;
                        /*создаем класс алгоритма алгоритм*/
                        createOneImageDetector();
                        /*создаем класс калибровщик*/

      createOneImageCalibrator(IMG_CALIBRATION_METHODS::KnownDiameter/*эту  величину
взять из настроек*/);
                        /*кубатурник не зависит от изображения его создаем один раз
в конструкторе*/

                  }
            }
            /*просим калибровщик заполнить данные*/
            oneImageCalibrator->RequestToExecute(gcnew MrUpdateResultData);
      }
            /*обработкик событий "программно изменилось отображаемое свойство"
              на эту функцию подписываются все объекты, которые отображают
              свойства в настройках*/
private: System::Void oneImageModel_PropertyChanged(Object^ sender,System::EventArgs^
e)
             {
                   /*генерировать сообщение изменения свойств*/
                   this->PropertyChanged(this,nullptr);
             }
#pragma endregion обработчики событий OneImage
#pragma region реализация интерфейса IModel
public:
      /*событие подмены объекта настроек (к этому классу неприминимо)*/
      virtual event ReplacePropertyHandler^ ReplaceProperty;
      /*событие изменились данные настроек*/
      virtual event System::EventHandler^ PropertyChanged;
      /*получить список объектов настроек*/
      virtual List<Object^>^ GetSettingObjects()
      {
            List<Object^>^ settingsObjects = gcnew List<Object^>();
            settingsObjects->Add(this);
            return settingsObjects;
      }
      /*запрос на выполнение запроса*/
      virtual System::Void RequestToExecute(IModelRequest^ modelRequest)
      {
            /*определить, что за запрос и выполнить или переслать тому объекту,
который исполнит*/
            switch ( modelRequest->ID )
            {
                  /*запросы к изображению*/
            case MODEL_REQUEST_ID::OPEN_IMAGE:
            case MODEL_REQUEST_ID::UPDATE_RESULT_DATA:
                  oneImage->RequestToExecute(modelRequest);
                  break;
                  /*запросы к алгоритму*/
            case MODEL_REQUEST_ID::RUN_ALGORITM:
            case MODEL_REQUEST_ID::ADD_OBJECT:
            case MODEL_REQUEST_ID::DELETE_OBJECT:
            case MODEL_REQUEST_ID::CHANGE_OBJECT:
```

```
                if(oneImageDetector != nullptr)
                {
                        /*пересылаем алгоритму*/
                        oneImageDetector->RequestToExecute(modelRequest);
                }
                else
                {
                        MessageBox::Show("Очень жаль, но изображение не
загружено");
                }
                break;
                /*запросы к калибровщику*/
            case MODEL_REQUEST_ID::ADD_CALIBRATE_OBJECT:
            case MODEL_REQUEST_ID::CHANGE_CALIBRATE_OBJECT:
            case MODEL_REQUEST_ID::DELETE_CALIBRATE_OBJECT:
                if(oneImageCalibrator != nullptr)
                {
                        /*пересылаем калибровщику*/
                        oneImageCalibrator->RequestToExecute(modelRequest);
                }
                else
                {
                        MessageBox::Show("Очень жаль, но изображение не
загружено");
                }
                break;
                /*запросы к кубатурнику*/
            case MODEL_REQUEST_ID::RECALCULATE:
                if(oneImageCubaturnic != nullptr)
                {
                        /*пересылаем калибровщику*/
                        oneImageCubaturnic->RequestToExecute(gcnew
MrUpdateResultData());
                }
                else
                {
                        MessageBox::Show("Очень жаль, но изображение не
загружено");
                }
                break;
            case MODEL_REQUEST_ID::FILL_REPORT:
                /*менеджер отчетов реализован вв другой сборке и не поддерживает
                  модель запросов и асинхронных операций Forest, поэтому все
проверки делаем сдесь*/
                if(AsyncOperation::InstanceOperation->IsBusy == false)
                {
                        if(reportManager != nullptr)
                        {
                                MrFillReport^ mrFillReport =
dynamic_cast<MrFillReport^>(modelRequest);
                                reportManager->CreateReport(mrFillReport-
>reportViewer,this->createReportSource());
                                delete mrFillReport;
                                mrFillReport = nullptr;
                        }
                }
                break;
            default:
                MessageBox::Show(this->ToString() + ": Не знаю, как выполнить
запрос " + modelRequest->ToString());
            }
```

```
	}
	/*событие "Запрос выполнен"*/
	virtual event System::EventHandler^ RequestExecuted;
	/*события об асинхронных операциях*/
	virtual event AsincNotificationEventHandler^ StartAsinc;
	virtual event AsincNotificationEventHandler^ StopAsinc;
#pragma endregion реализация интерфейса IModel
#pragma region реализация интерфейса IModelImagesStrategy
public:
	/*свойство стратегия*/
	[ReadOnlyAttribute(false)]
	[BrowsableAttribute(true)]
	[DisplayName("Стратегия")]
	[Description("Определяет, сколько изображений обрабатывается для штабеля ")]
	[Category("Модель")]
	[TypeConverter(EnumTypeConverter::typeid)]
	virtual property MODEL_IMAGE_STRATEGIES ImageStrategies
	{
		MODEL_IMAGE_STRATEGIES get()
		{
			return modelStrategy;
		}
		void set(MODEL_IMAGE_STRATEGIES val)
		{
			if(val != modelStrategy)
			{
				/*отправить сообщение о смене стратегии*/
				ImageStrategyChanged(val);
			}
		}
	}
	/*событие, об измененииии стратегии*/
	virtual event ImageStrategyEventHandler^ ImageStrategyChanged;
#pragma endregion реализация интерфейса IModelImagesStrategy
#pragma region свойства, отображающиеся в настройках
public:
	/*изображение*/
	[ReadOnlyAttribute(false)]
	[BrowsableAttribute(true)]
	[DisplayName("Изображение")]
	[Description("Изображение")]
	[Category("Свойства")]
	[Editor(OneImageDropDownEditor::typeid, UITypeEditor::typeid)]
	virtual property OneImage^ OneImage_
	{
		OneImage^ get()
		{
			return oneImage;
		}
		void set(OneImage^ val)
		{
		}
	}
	/*калибровка*/
	[ReadOnlyAttribute(false)]
	[BrowsableAttribute(true)]
	[DisplayName("Калибровка")]
	[Description("Предоставляет данные для калибровки изображения")]
	[TypeConverter(OneImageCalibrationMethodsType::typeid)]
	[Category("Свойства")]
	virtual property OneImageBaseCalibrator^ Calibrator
```

```
	{
		OneImageBaseCalibrator^ get()
		{
			return oneImageCalibrator;
		}
		void set(OneImageBaseCalibrator^ value)
		{
			if(value != nullptr)
			{
				/*применить калибровщик, выбранный пользователем*/
				createOneImageCalibrator(value->MethodKey);
			}
		}
	}
	[ReadOnlyAttribute(false)]
	[BrowsableAttribute(true)]
	[DisplayName("Вид")]
	[Description("Войства отображения графики")]
	[Category("Свойства")]
	[Editor(GraphicSettingsStyleEditor::typeid, UITypeEditor::typeid)]
	virtual property GraphicSettings^ GSettings
	{
		GraphicSettings^ get()
		{
			return GraphicSettings::GSettings;
		}
	}
	[ReadOnlyAttribute(false)]
	[BrowsableAttribute(true)]
	[DisplayName("Кубатурник")]
	[Description("Настройки расчета штабеля")]
	[Category("Свойства")]
	virtual property OneImageCubaturnic^ Cubaturnic
	{
		OneImageCubaturnic^ get()
		{
			return oneImageCubaturnic;
		}
	}
	[BrowsableAttribute(false)]
	virtual property Dictionary<IMG_CALIBRATION_METHODS,OneImageBaseCalibrator^>^
OneImageCalibrationMethods
	{
		Dictionary<IMG_CALIBRATION_METHODS,OneImageBaseCalibrator^>^ get()
		{
			return oneImageCalibrationMethods;
		}
	}
	/*получить результирующие данные*/
	[BrowsableAttribute(false)]
	property OneImageResultData^ ResultData
	{
		OneImageResultData^ get()
		{
			return oneImageResultData;
		}
	}
#pragma endregion свойства, отображающиеся в настройках
#pragma region переопределения базового класса
public:	virtual String^ ToString() override
	{
```

```
            return "Модель";
      }
#pragma endregion переопределения базового класса
#pragma region обработчики событий OneImageDetector
             /*запрос детектора выполнен*/
protected:     virtual     System::Void     oneImageDetector_RequestExecuted(Object^
sender,System::EventArgs^ e)
             {
                    /*просим калибровщик выпонить запрос "заполнить данные"
                     */
                    oneImageCalibrator->RequestToExecute(gcnew MrUpdateResultData);
             }
#pragma endregion обработчики событий OneImageDetector
#pragma region обработчики событий OneImageCalibrator
protected:     virtual     System::Void     oneImageCalibrator_RequestExecuted(Object^
sender,System::EventArgs^ e)
             {
                    /*просим кубатурник выпонить запрос "заполнить данные"*/
                    oneImageCubaturnic->RequestToExecute(gcnew MrUpdateResultData);
             }
#pragma endregion обработчики событий OneImageCalibrator
#pragma region обработчики событий oneImageCubaturnic
protected:     virtual     System::Void     oneImageCubaturnic_RequestExecuted(Object^
sender,System::EventArgs^ e)
             {
                    /*данные готовы уведомить форму*/
                    this->RequestExecuted(oneImageResultData,nullptr);
             }
#pragma endregion обработчики событий OneImageCubaturnic
#pragma region протектные методы
protected:
      /*создать салибратион метод*/
      void createOneImageCalibrator(IMG_CALIBRATION_METHODS calibrMethods_)
      {
            /*если текущий калибратор nullptr*/
            if(this->oneImageCalibrator == nullptr)
            {
                  /*создаем согласно IMG_CALIBRATION_METHODS
                    получаем значение по ключу*/
                  oneImageCalibrator          =          oneImageCalibrationMethods-
>default[calibrMethods_];
                  addOneImageCalibratorEvents();
            }
            else
            {
                  /*отписываемся от событий текущего*/
                  removeOneImageCalibratorEvents();
                  /*если изображение модели не равно изображению
                   калибровщика  создаем  заново  список  калибровщиков  с  новым
изображением*/
                  if(this->image != this->oneImageCalibrator->Image_)
                  {
                        createOneImageCalibrationMethods();
                  }
                  /*выбираем по ключу*/
                  oneImageCalibrator          =          oneImageCalibrationMethods-
>default[calibrMethods_];
                  /*подписываемся*/
                  addOneImageCalibratorEvents();
            }
      }
```

```
	/*подписаться на события калибратора*/
	void addOneImageCalibratorEvents()
	{
		if(this->oneImageCalibrator != nullptr)
		{
			oneImageCalibrator->RequestExecuted		+=		gcnew
System::EventHandler(this,&OneImageModel::oneImageCalibrator_RequestExecuted);
			oneImageCalibrator->PropertyChanged		+=		gcnew
System::EventHandler(this,&OneImageModel::oneImageModel_PropertyChanged);
		}
	}
	/*отписаться от событий калибратора*/
	void removeOneImageCalibratorEvents()
	{
		if(this->oneImageCalibrator != nullptr)
		{
			oneImageCalibrator->RequestExecuted		-=		gcnew
System::EventHandler(this,&OneImageModel::oneImageCalibrator_RequestExecuted);
			oneImageCalibrator->PropertyChanged		-=		gcnew
System::EventHandler(this,&OneImageModel::oneImageModel_PropertyChanged);
		}
	}
	/*создать массив калибровщиков*/
	void createOneImageCalibrationMethods()
	{
		clearOneImageCalibrationMethods();
		oneImageCalibrationMethods		=		gcnew
Dictionary<IMG_CALIBRATION_METHODS,OneImageBaseCalibrator^>();
		OneImageBaseCalibrator^ oibc;
		oibc = gcnew OneImageCutLogCalibrator(this->image,oneImageResultData);
		oneImageCalibrationMethods->Add(oibc->MethodKey,oibc);
		oibc = nullptr;
	}
	/*удалить список калибровщиков*/
	void clearOneImageCalibrationMethods()
	{
		if(oneImageCalibrationMethods != nullptr)
		{
			array<OneImageBaseCalibrator^>^	ioc_methods	=	gcnew
array<OneImageBaseCalibrator^>(oneImageCalibrationMethods->Count);
			oneImageCalibrationMethods->Values->CopyTo(ioc_methods,0);
			for(int i = 0;i< ioc_methods->Length;++i)
			{
				delete ioc_methods[i];
				ioc_methods[i] = nullptr;
			}
			Array::Clear(ioc_methods,0,ioc_methods->Length);
			oneImageCalibrationMethods->Clear();
			oneImageCalibrationMethods = nullptr;
		}
	}
	/*создаем кубатурник*/
	virtual void createOneImageCubaturnic()
	{
		clearOneImageCubaturnic();
		oneImageCubaturnic = gcnew OneImageCubaturnic(oneImageResultData);
		oneImageCubaturnic->RequestExecuted		+=		gcnew
System::EventHandler(this,&OneImageModel::oneImageCubaturnic_RequestExecuted);
		oneImageCubaturnic->PropertyChanged		+=		gcnew
System::EventHandler(this,&OneImageModel::oneImageModel_PropertyChanged);
	}
```

```
	/*удаляем кубатурник*/
	void clearOneImageCubaturnic()
	{
		if(oneImageCubaturnic != nullptr)
		{
			oneImageCubaturnic->RequestExecuted -= gcnew
System::EventHandler(this,&OneImageModel::oneImageCubaturnic_RequestExecuted);
			oneImageCubaturnic->PropertyChanged -= gcnew
System::EventHandler(this,&OneImageModel::oneImageModel_PropertyChanged);
			delete oneImageCubaturnic;
			oneImageCubaturnic = nullptr;
		}
	}
	/*удаляем oneImage*/
	virtual void clearOneImage()
	{
		if(oneImage != nullptr)
		{
			oneImage->RequestExecuted -= gcnew
System::EventHandler(this,&OneImageModel::oneImage_RequestExecuted);
			oneImage->PropertyChanged -= gcnew
System::EventHandler(this,&OneImageModel::oneImageModel_PropertyChanged);
			delete oneImage;
			oneImage = nullptr;
		}
	}
	/*создаем детектор*/
	void createOneImageDetector()
	{
		clearOneImageDetector();
		this->oneImageDetector = gcnew OneImageDetector(this-
>image,oneImageResultData);
		oneImageDetector->RequestExecuted += gcnew
System::EventHandler(this,&OneImageModel::oneImageDetector_RequestExecuted);
		oneImageDetector->PropertyChanged += gcnew
System::EventHandler(this,&OneImageModel::oneImageModel_PropertyChanged);
	}
	/*удаляем детектор*/
	void clearOneImageDetector()
	{
		if(this->oneImageDetector != nullptr)
		{
			oneImageDetector->RequestExecuted -= gcnew
System::EventHandler(this,&OneImageModel::oneImageDetector_RequestExecuted);
			oneImageDetector->PropertyChanged -= gcnew
System::EventHandler(this,&OneImageModel::oneImageModel_PropertyChanged);
			delete oneImageDetector;
			oneImageDetector = nullptr;
		}
	}
	/*создаем менеджер отчетов*/
	virtual void createReportManager()
	{
		this->clearReportManager();
		this->reportManager = gcnew ReportManager();
	}
	/*удаляем менеджер отчетов*/
	void clearReportManager()
	{
		if(reportManager != nullptr)
		{
```

```
                delete reportManager;
                reportManager = nullptr;
            }
        }
        /*создать данные, по которым строится отчет*/
        List<IReportData^>^ createReportSource()
        {
            List<IReportData^>^ source_ = gcnew List<IReportData^>();
            if(oneImage != nullptr)
            {
                source_->Add(oneImage);
            }
            if(oneImageCalibrator != nullptr)
            {
                source_->Add(oneImageCalibrator);
            }
            if(oneImageCubaturnic != nullptr)
            {
                source_->Add(oneImageCubaturnic);
            }
            if(oneImageResultData->CollectionData != nullptr)
            {
                source_->Add(oneImageResultData->CollectionData);
            }
            return source_;
        }
#pragma endregion приватные методы
};
}
```